MEMRİSTÖR TABANLI YAPAY SİNİR AĞI TASARIMI VE OPTİMİZASYONU

Baki GÖKGÖZ

Danışman: Doç. Dr. Tolga AYDIN

Doktora Tezi

Bilgisayar Mühendisliği Ana Bilim Dalı

2025

(Her hakkı saklıdır.)

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

MEMRİSTÖR TABANLI YAPAY SİNİR AĞI TASARIMI VE OPTİMİZASYONU

(Memristor Based Artificial Neural Network Design and Optimisation)

DOKTORA TEZİ

Baki GÖKGÖZ

Danışman: Doç. Dr. Tolga AYDIN

Erzurum

Ocak, 2025

KABUL VE ONAY TUTANAĞI

Baki GÖKGÖZ tarafından hazırlanan "MEMRİSTÖR TABANLI YAPAY SİNİR AĞI TASARIMI VE OPTİMİZASYONU" başlıklı çalışması 15/01/2025 tarihinde yapılan tez savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Bilgisayar Mühendisliği Ana Bilim Dalı, Bilgisayar Mühendisliği Bilim Dalında doktora tezi olarak kabul edilmiştir.

Jüri Başkanı:	Prof. Dr. Kemal BIÇAKCI *İstanbul Teknik Üniversitesi*	Aslı ıslak imzalıdır
Danışman:	Doç. Dr. Tolga AYDIN *Atatürk Üniversitesi*	Aslı ıslak imzalıdır
Jüri Üyesi:	Prof. Dr. İbrahim Yücel ÖZBEK *Atatürk Üniversitesi*	Aslı ıslak imzalıdır
Jüri Üyesi:	Doç. Dr. Ferhat BOZKURT *Atatürk Üniversitesi*	Aslı ıslak imzalıdır
Jüri Üyesi:	Dr. Öğr. Üyesi Özkan BİNGÖL *Gümüşhane Üniversitesi*	Aslı ıslak imzalıdır
İkinci Tez Danışmanı	Doç. Dr. Fatih GÜL *Recep Tayyip Erdoğan Üniversitesi*	Enstitü Yönetim Kurulunun …./…/…. tarih ve …… sayılı kararı.

Bu tezin Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği'nin ilgili maddelerinde belirtilen şartları yerine getirdiğini onaylarım.

Prof.Dr. Alper NUHOĞLU

Enstitü Müdürü

Aslı Islak İmzalıdır

Bu çalışma, Atatürk Üniversitesi BAP projeleri kapsamında desteklenmiştir.
Proje No: FDK-2022-9895

ETİK BİLDİRİM VE İNTİHAL BEYAN FORMU

Doktora Tezi olarak Doç. Dr. Tolga AYDIN danışmanlığında sunulan "MEMRİSTÖR TABANLI YAPAY SİNİR AĞI TASARIMI VE OPTİMİZASYONU" başlıklı çalışmanın tarafımızdan bilimsel etik ilkelere uyularak yazıldığını, yararlanılan eserlerin kaynakçada gösterildiğini, Fen Bilimleri Enstitüsü tarafından belirlenmiş olan Turnitin programı benzerlik oranlarının aşılmadığını ve aşağıdaki oranlarda olduğunu beyan ederiz.

Tez Bölümleri	**Tezin Benzerlik Oranı (%)**	**Maksimum Oran (%)**
Giriş	*2*	**30**
Kuramsal Temeller	*8*	**30**
Materyal ve Metot	*4*	**35**
Araştırma Bulguları ve Tartışma	*3*	**20**
Sonuçlar ve Öneriler	*0*	**20**
Tezin Geneli	*7*	**25**

Not: *Yedi kelimeye kadar benzerlikler ile Başlık, Kaynakça, İçindekiler, Teşekkür, Dizin ve Ekler kısımları tarama dışı bırakılabilir. Yukarıdaki azami benzerlik oranları yanında tek bir kaynaktan olan benzerlik oranlarının* ***%5'den*** *büyük olmaması gerekir.*

Sunulan bilgilerin doğru olduğunu, aksi halde doğacak hukuki sorumlulukları kabul ettiğimizi beyan ederiz.

Tez Yazarı (Öğrenci)	**Tez Danışmanı**
Baki GÖKGÖZ	**Doç. Dr. Tolga AYDIN**
15.1.2025	**15.1.2025**
İmza: Aslı ıslak imzalıdır	İmza: Aslı ıslak imzalıdır

***** Tez ile ilgili YÖKTEZ'de yayınlamasına ilişkin bir engelleme var ise aşağıdaki alanı doldurunuz.

☐ Tezle ilgili patent başvurusu yapılması / patent alma sürecinin devam etmesi sebebiyle Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 2 (iki) yıl süreyle engellenmiştir.

☐ Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 6 (altı) ay süreyle engellenmiştir.

TEŞEKKÜR

Çalışmalarım süresince bilgi ve deneyimleriyle bana rehberlik eden, değerli görüşlerini paylaşarak birçok konuda destek olan danışmanım Sayın Doç. Öğr. Tolga AYDIN'a en derin teşekkürlerimi sunarım.

Tez konusunun belirlenmesinde rehberlik eden, her zaman değerli zamanını ayırarak yardımlarını esirgemeyen ikinci danışmanım Sayın Doç. Dr. Fatih GÜL'e en içten teşekkürlerimi ve saygılarımı iletmek isterim.

Çalışmalarım boyunca bilgi ve deneyimleriyle bana yol gösteren, görüş ve destekleriyle katkı sağlayan Sayın Prof. Dr. İbrahim Yücel ÖZBEK'e ve Sayın Doç. Dr. Ferhat BOZKURT'a da teşekkür ederim.

Ayrıca, her zaman yanımda olan ve desteklerini hiçbir şekilde eksik etmeyen aileme minnettarım.

Lisansüstü eğitimim süresince bana destek veren ve Tez Projesi kapsamında (FDK-2022-9895) katkılarda bulunan Atatürk Üniversitesi BAP birimine de teşekkür ederim.

Baki GÖKGÖZ

ÖZET

DOKTORA TEZİ

MEMRİSTÖR TABANLI YAPAY SİNİR AĞI TASARIMI VE OPTİMİZASYONU

Baki GÖKGÖZ

Danışman: Doç. Dr. Tolga AYDIN
İkinci Tez Danışmanı: Doç. Dr. Fatih GÜL

Amaç: Bu çalışmanın amacı, memristör tabanlı nano-sinaptik cihazların yapay sinir ağı (YSA) uygulamalarında kullanılabilirliğini bilgisayar ortamında simüle etmek ve bu cihazların performansını detaylı bir şekilde değerlendirmektir. Simülasyon çalışmaları, cihazların doğruluk, enerji verimliliği, eğitim süresi, donanım kapasitesi ve ölçeklenebilirlik gibi kritik performans kriterleri açısından sağladığı katkıları ortaya koymayı hedeflemektedir. Bunun yanı sıra, bu cihazlar üzerinde kullanılan optimizasyon yöntemlerinin etkinliği ve bu yöntemlerin performans üzerinde oluşturduğu iyileştirmeler incelenecektir. Çalışmanın, hem yeni nesil sinaptik cihazların geliştirilmesine hem de yapay sinir ağı uygulamalarında optimizasyon yöntemlerinin rolünün daha iyi anlaşılmasına katkı sağlaması amaçlanmaktadır.

Yöntem: Bu çalışmada, memristör tabanlı nano-sinaptik cihazlar kullanılarak sinir ağlarının donanım üzerinde uygulanabilirliği incelenmiş ve bu yöntem çeşitli optimizasyon algoritmaları yardımıyla doğruluk, alan kullanımı ve enerji tüketimi açısından değerlendirilmiştir. Önerilen donanım tabanlı model, özellik tanıma ve sınıflandırma yeteneklerine sahip olup, elle yazılmış ve bilgisayar tarafından üretilmiş rakamların tanınmasında kullanılmaktadır. Model, ileri besleme (FF) ve geri yayılma (BP) süreçlerini entegre ederek giriş verilerini çıkışlarla eşleştiren bir yapı sunmaktadır. Eğitim süreci, FF ve BP olmak üzere iki aşamadan oluşmaktadır. İleri besleme aşamasında, girdi verileri sinir ağının giriş katmanına alınır, ardından ağırlıklı toplamlar ve aktivasyon fonksiyonları aracılığıyla gizli katmanlardan geçirilerek çıktı katmanına iletilir. Çıktılar, doğru etiketlerle karşılaştırılarak tahmin hatası hesaplanır. Geri yayılma aşamasında ise bu hata, sinir ağı boyunca geriye doğru yayılır ve ağırlıklar optimizasyon yöntemleri kullanılarak ayarlanır. Optimizasyonun temel amacı, modelin hem eğitim hem de test verilerinde yüksek doğruluk elde etmesini sağlamak, genelleme yeteneğini artırarak farklı veri setlerinde başarılı olmasını mümkün kılmak ve tüm bunları hesaplama maliyetlerini minimize ederek en verimli şekilde gerçekleştirecek en uygun yapılandırmayı bulmaktır.

Bulgular: Bu çalışma, memristör tabanlı memristörlerin nöromorfik uygulamalarda yüksek doğruluk, enerji verimliliği ve genelleme kapasitesi sunduğunu ortaya koymaktadır. CMOS uyumlu yapıları ve biyolojik sinapsları taklit edebilme özellikleri sayesinde, bu cihazlar enerji verimli donanımlar için önemli avantajlar sağlamaktadır. Optimizasyon algoritmaları arasında AdaDelta, doğruluk ve enerji verimliliği açısından en iyi performansı sergilemiştir. Modelin farklı veri kümelerindeki genelleme kapasitesi, Memristör tabanlı cihazların geniş bir makine öğrenimi yelpazesinde etkili bir şekilde kullanılabileceğini göstermektedir.

Sonuç: Memristör tabanlı sinaptik cihazlara dayalı sinir ağının performansı, MNIST ve CIFAR veri kümeleri üzerinde farklı optimizasyon yöntemleri kullanılarak kapsamlı bir şekilde değerlendirilmiştir. %90 doğruluk oranına ulaşan model, optimizasyon algoritmaları arasında sağlamlık ve genelleme kapasitesi sergilemiştir. Elde edilen sonuçlar, memristör tabanlı cihazların nöromorfik hesaplama ve donanım tabanlı yapay zekâ uygulamalarında, çeşitli optimizasyon yöntemleriyle birleştirildiğinde etkin bir çözüm sunduğunu ortaya koymaktadır. Ayrıca önerilen TiO_2 tabanlı model, Ag:Si tabanlı sinir ağına göre %20 daha az enerji, %16 daha yüksek doğruluk ve %18 daha düşük gecikme sağlamaktadır. Bu çalışma, gelecekte daha enerji verimli ve hassas sinir ağı modellerinin geliştirilmesine katkıda bulunacak önemli bulgular sağlamaktadır.

Anahtar Kelimeler: Derin öğrenme, makine öğrenimi, memristörler, nöromorfik hesaplama, optimizasyon algoritmaları, sinapslar, Memristör.

Ocak 2025, 149 sayfa

ABSTRACT

DOCTORAL DISSERTATION

MEMRISTOR BASED ARTIFICIAL NEURAL NETWORK DESIGN AND OPTIMISATION

Baki GÖKGÖZ

Supervisor: Assoc. Prof. Dr. Tolga AYDIN
Co-supervisor: Assoc. Prof. Dr. Fatih GÜL

Purpose: The aim of this study is to simulate the usability of memristor-based nano-synaptic devices in artificial neural network (ANN) applications in a computer environment and to evaluate the performance of these devices in detail. The simulation studies aim to reveal the contributions of these devices in terms of critical performance criteria such as accuracy, energy efficiency, training time, hardware capacity, and scalability. Additionally, the effectiveness of optimization methods applied to these devices and the improvements they bring to performance will be examined. The study is intended to contribute both to the development of next-generation synaptic devices and to a better understanding of the role of optimization methods in artificial neural network applications.

Method: In this study, the feasibility of implementing neural networks on hardware using memristor-based nano-synaptic devices was examined, and this method was evaluated in terms of accuracy, area usage, and energy consumption with the help of various optimization algorithms. The proposed hardware-based model possesses feature recognition and classification capabilities and is used for recognizing both handwritten and computer-generated digits. The model integrates feedforward (FF) and backpropagation (BP) processes, providing a structure that maps input data to outputs. The training process consists of two stages: FF and BP. In the feedforward stage, input data is fed into the input layer of the neural network, then passed through the hidden layers using weighted sums and activation functions, and finally delivered to the output layer. The outputs are compared with the correct labels to calculate prediction error. In the backpropagation stage, this error is propagated backward through the neural network, and the weights are adjusted using optimization methods. This process is performed faster and more efficiently than traditional gradient descent methods, aiming to minimize the error.

Finding: This study demonstrates that memristor-based memristors offer high accuracy, energy efficiency, and generalization capacity in neuromorphic applications. Due to their CMOS-compatible structures and ability to mimic biological synapses, these devices provide significant advantages for energy-efficient hardware. Among the optimization algorithms, AdaDelta exhibited the best performance in terms of accuracy and energy efficiency. The model's generalization capacity across different datasets highlights the potential for Memristör-based devices to be effectively utilized in a wide range of machine learning applications.

Result: The performance of the neural network based on memristor synaptic devices was comprehensively evaluated on the MNIST and CIFAR datasets using different optimization methods. The model achieved a 90% accuracy rate, demonstrating robustness and generalization capacity among the optimization algorithms. The results reveal that memristor-based devices, when combined with various optimization methods, offer an effective solution for neuromorphic computing and hardware-based artificial intelligence applications. Moreover, the proposed TiO_2 based model provides 20% less energy, 16% higher accuracy and 18% lower latency than the Ag:Si based neural network. This study provides significant findings that will contribute to the development of more energy-efficient and precise neural network models in the future.

Keywords: Deep learning, machine learning, memristors, neuromorphic computing, optimization algorithms, synapses, Memristör.

January 2025, 149 pages

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

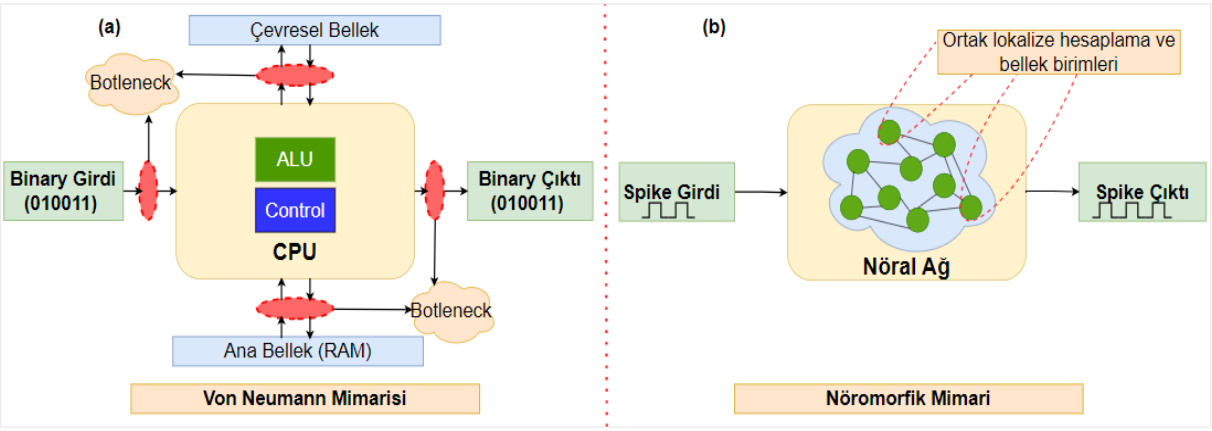
KISALTMALAR ve SİMGELER DİZİNİ

AI : Artificial Intelligence
DNN : Deep Neural Network (Derin Sinir Ağları)
CMOS : Complementary Metal Oxide Semiconductor (Bütünleyici Metal Oksit Yarı İletken)
VN : Von Neumann
BT : Bilgisayar Teknolojileri
CPU : Central Processing Unit (Merkezi İşlem Birimi)
GPU : Graphics Processing Unit (Grafik İşlem Birimi)
SGD : Stochastic Gradient Descent (Stokastik Gradyan İnişi)
LTP : Long-Term Potentiation (Uzun Süreli G STP üçlenme)
LTD : Long-Term Depression (Uzun Süreli Depresyon)
STP : Short-Rerm Potentiation (Kısa Süreli Potansiyasyon)
RRAM : Resistive Random-Access Memory (Dirençli Rastgele Erişimli Bellek)
ANN : Artificial Neural Network (Yapay Sinir Ağları)
DL : Deep Learning (Derin Öğrenme)
ANN : Artificial Neural Network (Yapay Sinir Ağları)
RNN : Recurrent Neural Network (Tekrarlayan Sinir Ağı)
LSTM : Long Short Term Memory (Uzun Kısa Süreli Bellek)
CF : Critical Fault (Kritik Hatalar)
WTA : Winner-Take-All
SNN : Spike Neural Network (Spike Sinir Ağları)
FPGAs : Field Programmable Gate Arrays (Programlanabilir Kapı Dizileri)
BP : Backpropagation (Geriyayılım)
FF : Feedforward (İleriyayılım)
FFNN : Feedforward Neural Network (İleri Beslemeli Yapay Sinir Ağı)
ACC : Accuracy (Doğruluk)
MLP : Multi Layer Perceptron (Çok Katmanlı Algılayıcı)
PCM : Phase Change Memory (Faz Değişimi Belleği)
CBM : Conductive Bridge Memristors (İletken Köprü Memristörleri)
HP : Hewlett Packard
STDP : Spike Time-Dependent Plasticity (Spike Zamanına Bağlı Plastisite)
GRU : Gated Recurrent Unit (Geçitli Tekrarlayan Birim)
ACU : Approximated Computation Unit (Yaklaşık Hesaplama Birimi)
ASIC : Application-Specific Integrated Circuit (Özel Entegre Devreler)
TPU : Tensor Processing Unit (Tensör İşleme Birimi)
YSA : Yapay Sinir Ağları

GİRİŞ

Yapay zekâ kullanımının hızla yaygınlaştığı günümüz bilgi teknolojileri dünyasında, makine öğrenimi ve derin öğrenme hızla yaygınlaşmakta ve çeşitli uygulama alanları bulmaktadır (Ren et al., 2022). Akademik dünyada keşfedilip kullanılmaya başlanan bu yöntemler, hızla büyüyerek hem akademik hem de endüstriyel uygulamalarda önem kazanmıştır. Aynı zamanda veri biliminin de vazgeçilmez bileşenleridir (Sun et al., 2020). Gelişen bilgisayar teknolojileri sayesinde yapay zekâ alanlarında da önemli bir yer edinmişlerdir (S. Kim et al., 2019). Bununla birlikte makine öğrenimi ve derin öğrenme yapısında kullanılan, Derin Sinir Ağları (DNN) yüksek doğruluk, mükemmel ölçeklenebilirlik ve kendini uyarlama özellikleri nedeniyle yoğun olarak kullanılmaktadır (Ren et al., 2022). Bu yoğun kullanımın bir sonucu olarak, DNN modelleri daha büyük ve daha derin ağ yapıları oluşturacak şekilde geliştirilmektedir. Ayrıca, yapay zeka uygulamalarında kullanılan derin öğrenme algoritmaları süreçleri otomatikleştirmekte, verileri analiz etmekte ve tahmin işlevlerini yerine getirmektedir (Bengio et al., 2013).

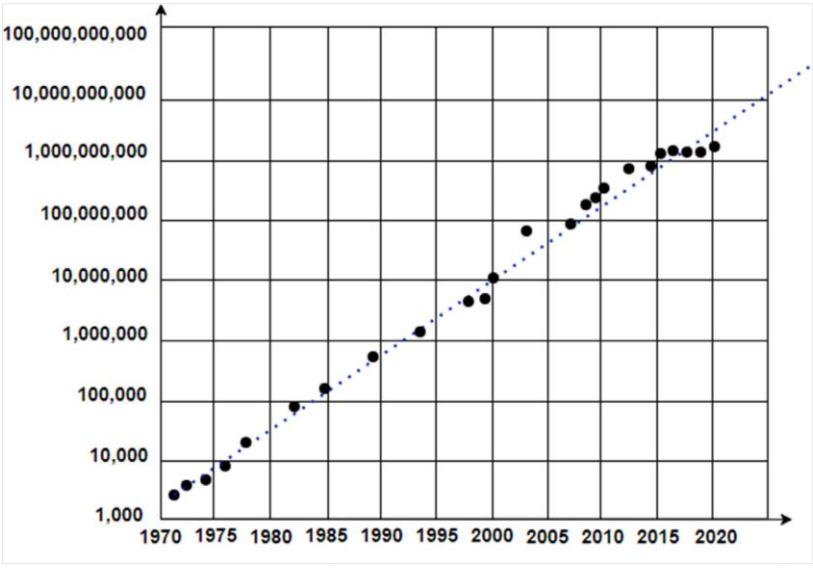
DNN modellerinin doğası gereği yüksek hesaplama gücü ve geniş bellek depolama alanı gerektirmektedir. Bu gereksinimler karşısında Von Neumann mimarisine dayalı klasik bilgi işleme ve hesaplama yöntemleri yetersiz kalmaktadır (S. Yu et al., 2011). Bu yetersizlik, bilgi işleme yapılarının biyolojik sinir sistemlerinden oldukça farklı olmasından kaynaklanmaktadır. Şekil 1(a) sıralı işlemede hesaplama ve belleğin ayrı olduğunu gösterirken, Şekil 1(b) paralel işlemede hesaplama ve belleğin bütünleşik olduğunu ve işlemlerin eş zamansız olarak gerçekleştiğini göstermektedir.



Şekil 1. (a) ve (b) Von Neumann mimarisi ile nöromorfik mimari arasındaki karşılaştırmayı göstermektedir. Bu iki mimari, burada gösterildiği gibi çalışma, organizasyon, programlama, iletişim ve zamanlama açısından temelde farklılık göstermektedir.

VN mimarisine dayalı geleneksel bilgisayar sistemleri iyi tanımlanmış yapısal problemlerin çözümünde etkilidir. Ancak derin öğrenme ve yapay zeka gibi biyolojik sistemleri taklit eden yöntemlerin kullanılması için uygun değildir (S. Yu et al., 2011). Bunun yerine bu tür yöntemler için daha karmaşık mimariler kullanılmalıdır. Bunun nedeni, mantık tabanlı donanım ve yazılım sistemleri mimarisinin biyolojik sinir sisteminden önemli ölçüde farklı olmasıdır (B. Gao et al., 2016). Biyolojik beyin, öğrenme işlevini milyonlarca sinapsın paralel çalışmasıyla gerçekleştirmektedir (Kuzum et al., 2013). Örneğin, karmaşık bir yapay zeka probleminin çözümünde kullanılan bir süper bilgisayar enerji maliyeti açısından 1 megawattan fazla güce ihtiyaç duyarken, insan beyni çalışma mimarisi sayesinde toplamda yaklaşık 10 watt'lık enerji tüketmektedir (Kuzum et al., 2013). Bu durum, klasik sistemlerin enerji gereksinimlerinin uygulama maliyetleri bakımından önemli bir zorluk teşkil ettiğini göstermektedir (Cumming et al., 2014). Enerji maliyeti sorununa ek olarak, klasik yöntemlerde verilerin sürekli olarak işlemci ve bellek birimleri arasında taşınması gerekmekte, bu da veri yoğun uygulamalarda gecikmelere yol açmaktadır (Eryilmaz *et al.*, 2014; Mutlu *et al.*, 2019). Bu gecikmeler özellikle gerçek zamanlı uygulamalarda temel bir sorun olarak ortaya çıkmaktadır (Eryilmaz et al., 2014; Gul, 2020; Sebastian et al., 2020). Bu sorunların üstesinden gelmek için memristör benzeri cihazlar gibi özelleşmiş, konuma özgü donanım birimlerinin oluşturulması düşünülmektedir (Sung, Hwang and Yoo, 2018).

Donanım hızlandırma yoluyla Derin Sinir Ağlarının (DNN) performansını ve enerji optimizasyonunu artırmak için hem akademik hem de endüstriyel sektörlerde kapsamlı araştırmalar yürütülmektedir.

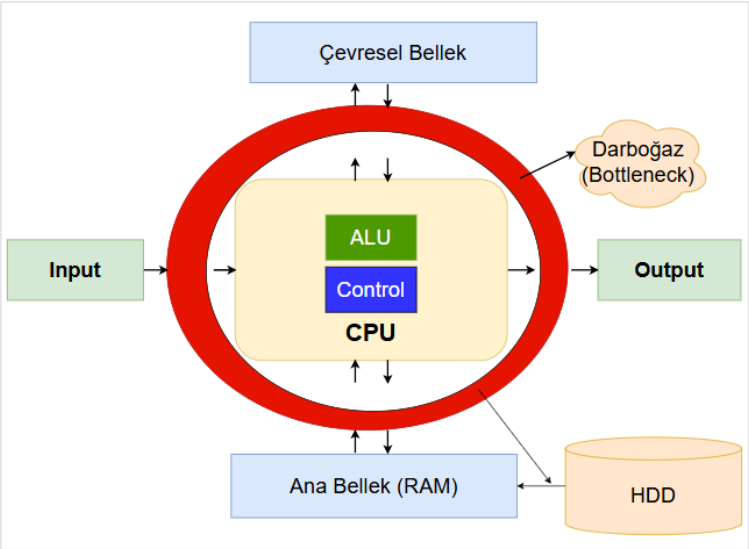


Şekil 2. Moore Yasası, işlemci başına transistör sayısındaki artışı ifade eder (1971-2020).

Geçtiğimiz elli yılda, Moore Yasası, Dennard Ölçeklendirme ve Von Neumann mimari trendlerinin etkili birleşimi ve kesişimi sayesinde bilgi teknolojisi alanında hızlı ve etkili bir dönüşüm yaşanmıştır (Tsai et al., 2018). Moore Yasası, transistör başına maliyetin üstel olarak azalmasını tanımlamaktadır (Moore, 1965). Bu trend, Şekil 2'de ifade edildiği gibi yıllar içindeki değişim grafiğinde gösterilmektedir. Bir dizi "ölçeklendirme yasası" (Dennard et al., 1974) ile tanımlanan Dennard ölçeklendirmesi, daha küçük transistörlerin daha hızlı çalışmasını ve aynı zamanda daha az güç tüketmesini sağlar.

Von Neumann mimarisinin çok yönlülüğü, geliştiricilerin CPU ve GPU'ları modüler bileşenler olarak kullanarak çeşitli karmaşık bilgi işlem sistemleri oluşturmalarına olanak sağlamaktadır (Tsai et al., 2018b). Son yıllarda, güçlü bir etkiye sahip olabilecek bu eğilimlerin kesişimi geçerliliğini büyük ölçüde yitirmeye başlamıştır (Radack & Zolper, 2008). Cihazların ihtiyaç duyduğu güç ve voltaj parametrelerindeki değişkenlik nedeniyle, cihaz ölçeklendirme işlevleri daha zorlu hale gelmiştir (Zidan et al., 2018). Bu zorluklar, cihazların optimum işlevselliğe ulaşmasını karmaşık ve zor hale getirmektedir (Schuman et al., 2017). Sonuç olarak, bellek ve işlemci arasındaki veri aktarımı için harcanan zaman ve enerji, gerçek zamanlı görüntü tanıma ve doğal dil işleme gibi yoğun veri işleyen ve analiz eden sistemler için sorunlu hale gelmiştir (Eryilmaz et al., 2014; Kuzum et al., 2013).

Günümüzde kullanılan hesaplama sistemleri Von Neumann mimarisi üzerine inşa edilmiştir (del Valle et al., 2018). Bu mimaride veriler önce merkezi işlem birimine (CPU) taşınmakta, burada işlenmekte, ardından işlenen veriler ana belleğe veya diğer bellek birimlerine aktarılmakta ve işlemler bu şekilde döngüsel olarak devam etmektedir (H. Li et al., 2015). Veri hacminin yüksek olduğu durumlarda, ilgili verilerin aktarılması ve işlenmesi zaman ve enerji açısından önemli bir maliyet sorunu oluşturmaktadır (Kvatinsky et al., 2014). Ayrıca, verilerin bellek bileşenlerinden alınmasına bağlı gecikme, özellikle yapay zeka alanında artan iş yükleriyle uğraşan çeşitli uygulamalarda performans açısından önemli bir engel oluşturmaktadır (Sebastian *et al.*, 2020). Ayrıca Şekil 3'te belirtildiği gibi sistemin veri transferi olan her yerinde darboğaz (bottleneck) problemi ortaya çıkmaktadır.



Şekil 3. Von Neumann mimarisinde darboğaz (bottleneck) problemi

Bilim insanları, Von Neumann mimarisinin doğasında bulunan darboğaz sorununu çözmek için aktif olarak hem donanım hem de yazılım yaklaşımları geliştirmektedirler (Drakopoulos et al., 2021; Xia et al., 2021) Yöntemlerden biri ve şu anda en popüler olanı, biyolojik beynin çalışma modelinden ilham alan nöromorfik sistemlerdir (Eshraghian, Wang and Lu, 2022). Biyolojik beynin yapısı incelendiğinde, hesaplama için düşük enerji gerektiren çoklu nöronları, uyarlanabilir bellek birimleri olarak hizmet veren sinapslarla birbirine bağlayan oldukça paralel çalışan bir mimari ile karakterize edilir (Cao et al., 2023).

Yapay Sinir Ağı (YSA) ve Nöromorfik Hesaplama

YSA, ham veri sınıflandırması ve örüntü tanımayı içeren görevlerde modern geleneksel Von Neumann işlemcilerden daha iyi performans gösterebilmektedir. İnsan beyninin tasarımından esinlenen nöromorfik donanım sistemleri (Indiveri et al., 2011), geleneksel Von Neumann mimarisinden farklı olarak güçlü ve verimli bilgi işleme yetenekleri sağlama kapasitesine sahiptir. Bu tür bir hesaplama sistemi hatalara karşı dirençli, son derece paralel ve enerji tasarrufludur (Mead, 1990). Bununla birlikte, günümüzde "nöromorfik hesaplama" olarak adlandırılan farklı uygulama ve tasarımların sayısı önemli ölçüde artmış ve tamamen farklı yaklaşımları kapsar hale gelmiştir (Nawrocki et al., 2016). Bu çalışmaların büyük çoğunluğu beynin birkaç nöronunu taklit etmeyi amaçlayan donanım tabanlı yeni cihazları içerirken, sistemin bir diğer kısmı da kısmen veya tamamen beynin çalışma modelinden esinlenen yeni yazılım algoritmalarını içermektedir (Lukoševičius & Jaeger, 2009). Sonuç olarak memristör devre elemanları kullanılarak donanım tabanlı sistemlerin geliştirilmesi üzerine çalışmalar yapılmıştır.

Memristör (L. Chua, 1971), değişken bir direnç gibi davranarak direnç değerini değiştiren ve akımın geçişi sırasında kapasitif bir eleman gibi davranarak akımın akışını kontrol eden bir devre elemanıdır (Oli-Uz-Zaman et al., 2022). Yeni bir donanım teknolojisi olan memristörler çok küçük ölçeklerde çalışmakta ve büyük ölçekli entegrasyonlar için kullanılabilmektedir (Gharpinde et al., 2018). Memristörlerin operasyonel mantığı ile karşılaştırıldığında, klasik tamamlayıcı metal-oksit-yarı iletken (CMOS) çalışma yöntemi, küçük alan gereksinimleri ve düşük güç tüketimi gibi benzer avantajlar sergilemektedir (Cheng & An, 2021a).

Memristör tabanlı derin sinir ağlarının avantajlarına rağmen, cihaz değişkenliği, mevcut teknolojinin sınırlamaları ve memristörlerin esnekliği gibi ele alınması gereken zorluklar vardır. Bu sorunların uygulamaların işleyişi üzerinde olumsuz etkileri bulunmaktadır. Sonuç olarak, donanım tabanlı sinir ağı uygulamalarında üstesinden gelinmesi gereken zorluklar vardır. Bu zorlukları aşmak için memristörler diğer sistemlere kıyasla daha fazla potansiyel çözümü sağlayabileceği yapılan çalışmalarla ortaya konulmuştur (Cheng & An, 2021).

Memristör tabanlı donanım sistemleri üzerine çeşitli inceleme makaleleri yapılmıştır. Mutlu ve meslektaşları, derin sinir ağlarını (DNN'ler) verimli bir şekilde işlemeyi amaçlayan son gelişmeler üzerine kapsamlı bir çalışma (Mutlu *et al.*, 2019) yürütmüştür.

Sebastian ve ekibi, bellek aygıtları tarafından etkinleştirilen temel hesaplama işlevlerinin yanı sıra bilimsel hesaplama, sinyal işleme, optimizasyon, makine öğrenimi, derin öğrenme ve stokastik hesaplama gibi uygulamaları inceleyen bir çalışma (Sebastian *et al.*, 2020) sunmuştur.

Bir diğer çalışma, Von Neumann mimarisindeki verimsizlikleri ortadan kaldırmak için farklı fiziksel prensiplere dayanan dirençli anahtarlama malzemelerinin tasarımına odaklanmıştır. Wang ve ekibi, dirençli anahtarlama süreçlerine (RSM) yol açan fiziksel mekanizmalarla ilgili çalışmaları incelemiş ve mekanizmaların temsil kabiliyeti, anahtarlama hızı, enerji, güvenilirlik ve cihaz yoğunluğu hakkında bilgi vermiştir (Z. Wang et al., 2020). Yang ve meslektaşları, derleme makalelerinde çeşitli memristör sinaptik cihazları uyaran modellerine göre sınıflandırmış ve bu sinaptik cihazların çalışma mekanizmalarının ayrıntılı bir analizini yapmıştır. Bir başka çalışma (S. Chen et al., 2023) ile hem biyolojik hem de yapay hücrelerde nöral sinyal üretimi ve iletiminin karmaşık süreçlerine yönelik mevcut yaklaşımların genel bir incelemesi sunulmuştur.

Günümüzde doğal dil işleme, metin tahmini, nesne algılama, konuşma ve görüntü tanıma gibi çeşitli yapay zekâ uygulamalarında YSA mimarilerinin kullanımı önemli ölçüde artmıştır. Geleneksel YSA uygulamalarında, bellek ve işlem birimleri arasında büyük ölçekli veri

hareketleri gerçekleşmekte ve bu işlemler yüksek hesaplama maliyetleri gerektirmektedir. Bu hesaplama süreçlerini daha verimli hale getirmek amacıyla çeşitli yazılım ve donanım çözümleri geliştirilmektedir. Ancak, tüm bu çabalara rağmen, veri trafiğindeki gecikme ve yüksek enerji tüketimi gibi Von Neumann mimarisinin darboğaz sorunları devam etmektedir. Bu darboğazın üstesinden gelmek için yapay zekâ uygulamalarına özel donanım elemanlarının tasarlanması ve geliştirilmesi gerekmektedir. Bu doğrultuda, nörobiyolojik sistemlerden esinlenen çeşitli özelliklerin donanım düzeyinde tasarlanması ve entegre edilmesi, bu soruna etkili bir çözüm sunabilir. Bu yaklaşım çerçevesinde, özellikle memristör tabanlı nöromorfik hesaplama sistemleri dikkat çekmektedir. Memristörler, uçucu olmayan bellek özellikleri ve analog davranışları sayesinde hız ve enerji verimliliği açısından umut verici donanım iyileştirmeleri sunmaktadır. Sinaptik ağırlıkların etkin bir şekilde depolanmasına ve işlenmesine olanak tanıyan bu cihazlar, donanım düzeyinde performans artırıcı çözümler sunmaktadır.

Bu çalışmada, güncel sistemlerden farklı olarak doğrudan derin öğrenme işlevlerini yerine getirebilen ve biyolojik beyin yapısını taklit edebilen memristör tabanlı nöromorfik hesaplama sistemleri ele alınmaktadır. Bu bağlamda, çalışmada derin öğrenme ve makine öğrenmesi uygulamalarında sıkça kullanılan Stokastik Gradyan İnişi (SGD) ve momentum tabanlı optimizasyon varyantlarının memristör tabanlı sistemler üzerindeki performansı kapsamlı bir şekilde deneysel olarak incelenmiştir. Ayrıca, nano ölçekli memristör tabanlı sinaptik cihazların öğrenme özellikleri, enerji verimliliği ve doğruluk oranları gibi önemli performans metrikleri detaylı olarak araştırılmıştır. Uygulamada MNIST ve CIFAR-10 veri setleri kullanılarak elde edilen deneysel sonuçlar sonraki bölümlerde paylaşılmıştır.

Tezin Amacı ve Alana Katkıları

Literatür incelemesi, memristör ileri teknoloji malzemeler kullanılarak çip üzerinde öğrenme süreçlerini gerçekleştirmeyi hedefleyen dirençli sinaptik diziler üzerine yapılan araştırmaları kapsamaktadır. Bu malzemeler, özellikle hafif yapıları ve enerji tasarrufu sağlayan özellikleri ile modern nöromorfik hesaplama sistemlerinin geliştirilmesinde etkili bir potansiyel sunmaktadır. Araştırmalar, memristör tabanlı sinaptik dizilerin elektriksel karakteristiklerini inceleyerek bu malzemelerin sinaptik davranışları nasıl taklit ettiğini anlamayı amaçlamaktadır. Bu çalışmalar, nöromorfik uygulamalar için gerekli olan doğrusal olmayan iletim, yüksek anahtarlama hızları, geniş bellek pencereleri, uzun süreli bellek tutma yetenekleri gibi birçok temel elektriksel özelliği analiz etmektedir. Bu bağlamda, özellikle sinaptik ağırlıkların sürekli olarak güncellenmesine olanak tanıyan çok katmanlı yapılar ve malzemelerin sinaptik

plastikliği, uzun süreli potansiyasyon (LTP) ve depresyon (LTD) gibi biyolojik fonksiyonları destekleyebilme kapasitesi ön plana çıkmaktadır.

Özetle literatür incelemesinde genellikle memristör malzemeleri kullanılarak çip üzerinde öğrenmeyi sağlamak amacıyla dirençli sinaptik dizilerin incelendiği görülmüştür. Bu çalışmaların odak noktası genellikle bu malzemelerin elektriksel özellikleri ve sinaptik davranışları olmuştur. Bu çerçevede, bu çalışma ile:

1. Memristör tabanlı nano-sinaptik bir cihazın yapay sinir ağı uygulamalarında kullanımını bilgisayar ortamında simüle ederek, bu cihaz üzerindeki optimizasyon yöntemlerinin performansını derinlemesine analiz ederek alana önemli katkılar sunulmuştur.
2. Bu çalışmada, yapay sinir ağı uygulamalarında kullanılan memristör tabanlı nano-sinaptik cihazın doğruluk, enerji optimizasyonu ve eğitim süresi gibi kritik performans ölçütleri optimize edilerek test edilmiştir.
3. MNIST ve CIFAR-10 veri kümeleri üzerinde gerçekleştirilen rakam tanıma ve sınıflandırma uygulamaları ile cihazın ve kullanılan optimizasyon yöntemlerinin performansı kapsamlı bir şekilde analiz edilmiştir.
4. Bu detaylı analizlerin sonucunda, cihazın kullanılan veri kümelerinde yüksek doğruluk oranları elde ettiği, dolayısıyla uygulamalarda hata oranını düşürerek etkili sonuçlar sunduğu görülmüştür.
5. Memristör tabanlı nano-sinaptik cihaz, enerji tüketimi bakımından klasik bilgisayar sistemlerine göre önemli ölçüde tasarruf sağlamakta, düşük enerji tüketimiyle yapay zekâ uygulamalarında yeni bir donanım çözümü olarak öne çıkabileceği gösterilmiştir.
6. Yapılan simülasyonlarda, Stocastic Gradient Descent (SGD) ve onun varyantları gibi optimizasyon yöntemleri kullanılarak cihazın dayanıklılığı, adaptasyon yeteneği ve farklı veri kümelerine uyum kabiliyeti test edilmiştir.
7. Bu çalışma, önerilen sistemin sağlamlığını ve uyarlanabilirliğini vurgulayarak, cihazın değişen koşullara ve parametre ayarlarına hızla adapte olabilen bir yapıya sahip olduğunu ortaya koymaktadır.
8. Nöromorfik mimarilerin potansiyeli ve yapay zekâ sistemlerine sağladığı katkılar ele alınarak, geleneksel bilgi işlem paradigmasının sınırlamalarının ötesine geçebilecek alternatif bir yol gösterilmiştir.

9. Elde edilen simülasyon sonuçları, memristör tabanlı nano-sinaptik cihazların, nöromorfik hesaplama sistemleri için sürdürülebilir ve yüksek performanslı bir çözüm sunduğunu, böylece yapay sinir ağı tabanlı uygulamalarda gelecekte yaygın bir kullanım alanına sahip olabileceğini göstermektedir.
10. Çalışmanın sonuçları, klasik bilgisayar sistemlerine bir alternatif olarak memristör tabanlı nano-sinaptik cihazların etkili bir donanım çözümü sunduğunu kanıtlamış olup, bu cihazların daha geniş alanlarda uygulanabilirliğini araştırmak için gelecekteki çalışmalara rehberlik etmektedir.

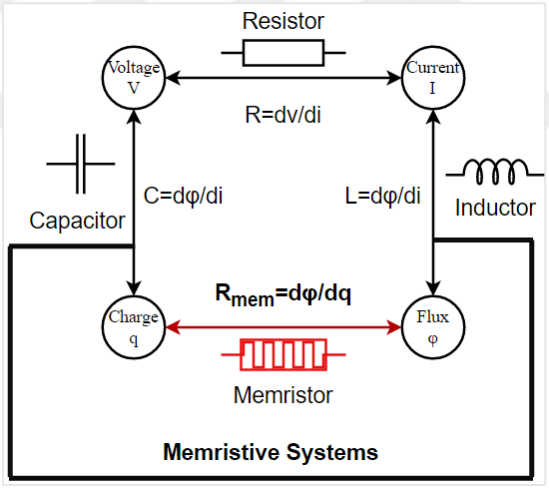
Bu çalışma, memristör tabanlı nano-sinaptik cihazların yapay sinir ağı uygulamalarındaki kapasitesini ve verimliliğini simülasyonlar aracılığıyla inceleyerek, optimizasyon yöntemlerinin bu cihazlar üzerindeki yansıması bilgisayar üzerinde simüle ederek detaylı bir biçimde değerlendirmektedir.

Bu tezin devamı, aşağıdaki şekilde yapılandırılmıştır. İlk olarak, Kuramsal Temeller bölümü, ilgili literatürü kapsamlı bir biçimde ele alarak tez konusuna ilişkin temel kavramları ve kuramsal çerçeveyi sunmaktadır. Ardından, Materyal ve Metot bölümünde, araştırmada kullanılan materyallerin özellikleri ayrıntılı olarak tanımlanmış ve önerilen yöntemin uygulama süreci açıklanmıştır. Bulgular ve Tartışma bölümünde ise araştırma sonuçları kapsamlı bir biçimde analiz edilerek, literatürdeki benzer çalışmalarla kıyaslanmış ve farklı açılardan yorumlanmıştır. Son olarak, Sonuçlar ve Öneriler bölümünde elde edilen bulgular ışığında bu alanda gelecekte yapılacak çalışmalara yönelik öneriler sunularak ve çalışmanın genel katkısı tartışılarak tez tamamlanmaktadır.

KURAMSAL TEMELLER

Memristörün Tarihsel Gelişimi

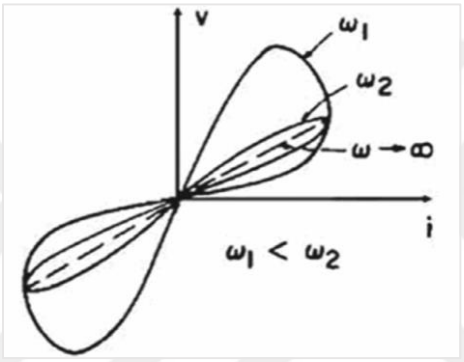
Ewald Georg von Kleist 1745 yılında kondansatörü, Georg Simon Ohm 1827 yılında direnci ve Michael Faraday 1831 yılında indüktörü keşfetmiştir (Mazumder et al., 2012). Bu devre elemanları ve aralarındaki ilişki Şekil 4'te gösterilmiştir. Şekil 4'te gösterilen devrede kondansatör elektrik yükü ile elektrik potansiyeli arasındaki bağlantıyı, direnç elektrik akımı ile voltaj arasındaki ilişkiyi, indüktör ise manyetik akı ile elektrik akımı arasındaki ilişkiyi kurmaktadır (R. Lin et al., 2023). Bununla birlikte, uzun bir süre boyunca akım ve yük arasındaki ilişkiyi kuran devre elemanı tanımlanmamıştır (Strukov et al., 2008). Leon Chua 1971 yılında simetri argümanlarına dayanarak dirençler, kapasitörler ve indüktörlerin yanında dördüncü bir devre elemanı olması gerektiğini savunmuş ve bu elemana hafızalı direnç olan memristör adını vermiştir (L. Chua, 1971). Chua ve Kang tarafından 1976 yılında yapılan başka bir çalışma (L. O. Chua & Sung Mo Kang, 1976) ile memristör daha da detaylandırılmış ve memristör özellikleri gösteren devre elemanları detaylı bir biçimde ele alınmıştır.



Şekil 4. Dört temel iki terminalli pasif devre elemanı

Çalışma, donanım tasarımının memristör kullanımını içerdiği "memristif sistemler" adı verilen yeni bir sistemi tanıtmakta ve böyle bir sistem için gerekli kriterleri belirlemektedir (Corinto et al., 2015). Ayrıca bu çalışma, memristörün akım-gerilim karakteristiğinin bir histerezis eğrisi sergilediğini tespit eden ilk çalışmadır (F. Y. Wang, 2008). Buna ek olarak, çalışma, yüksek frekanslarda bir direncin akım-gerilim karakteristiğinin doğrusal bir direncinkine benzer olduğunu açıklamaktadır (Adhikari et al., 2013; Sah et al., 2015). Şekil 5, Leon Chua'nın akım-gerilim karakteristiğine dayalı olarak tahmin edilen memristör histerezis eğrisini (Lyscaous)

göstermektedir. Chua ve Kang, çalışmalarında Hodgkin Huxley Sinir Modeli (Y. Liu et al., 2021) gibi sistemlerin memristörler kullanılarak uygulanabileceğini öne sürmüşlerdir. (L. O. Chua & Sung Mo Kang, 1976). Ayrıca, çalışmalarında memristif sistemler için modelleme yöntemlerini de göstermişlerdir. 2010 yılında, nöro plastisiteye benzeyen uzun vadeli plastisite (LTP), memristörlerin en yaygın olarak uygulanan temel işlevlerinden biri olan sürüklenme temelli memristörlerde (drift memristors) taklit edildi (Jo et al., 2010). 2010 yılında, biyolojik sistemlere kıyasla bağlantı ve işlev yoğunluğu sunan bir memristör çapraz çubuk dizisi de geliştirilmiştir (C. Li et al., 2019). Faz Değişim Belleği (PCM) olarak bilinen memristör türleri 2010 yılında uygulamalar için önerilmiştir (Wong et al., 2010). 2012 yılında, PCM tabanlı devreler nöromorfik spiking fonksiyonları taklit edilerek uygulanmıştır (Pickett et al., 2013).



Şekil 5. Leon Chua'nın öngördüğü memristör histerezisi (L. O. Chua & Sung Mo Kang, 1976)

2016 yılında Wang ve arkadaşları tarafından, metal iyonlarının redoks reaksiyonu ve metal iletken bir filament oluşumu yoluyla iletkenlik anahtarlamasını kontrol eden iletken köprü memristörleri (CBM) önerilmiştir. 2019 yılında, tamamen entegre ve programlanabilir memristör çip setleri geliştirilmiştir (Vaughan, 2023). Bu araştırma, bir memristör çapraz çubuk dizisi ile tamamlayıcı metal-oksit-yarı iletken (CMOS) kontrol devrelerinin kombinasyonunu içeriyordu. Sonuç olarak yüksek verimlilikle çarpma-biriktirme işlemlerini gerçekleştirebilen nöromorfik bir bilgi işlem çipinin oluşturulmasıydı (F. Cai et al., 2019). 2020'de yapılan bir çalışma, geleneksel çapraz çubuk dizilerinin iki boyutlu yapısını geliştirerek üç boyutlu memristör devreleri geliştirmeyi amaçlamıştır (P. Lin et al., 2020). Bellekte hesaplama yapabilen, CMOS ve dirençli rastgele erişimli bellek (RRAM) yapısına sahip memristör cihazları içeren, yüksek enerji verimli ve düşük gecikmeli bir donanım yapısı gerçekleştirilmiştir (Hung et al., 2021). 2022 yılına gelindiğinde, çip üzerinde iletişim için memristör çapraz çubuk dizilerine dayalı bir cihaz geliştirilmiş ve paralel veri işleme için kullanılmıştır (Choi et al., 2022).

Uzun yıllar boyunca memristörün teorik bir unsur olduğuna ve böyle bir devre elemanının gerçekte var olmadığına inanılıyordu (Johnsen, 2012; Meuffels & Soni, 2012). Bu algı nedeniyle araştırmacılar, 2008 yılında Hewlett Packard (HP) laboratuvarlarında fiziksel olarak uygulanana kadar bu yeni devre elemanına fazla ilgi göstermemiştir (Strukov et al., 2008). Memristörün HP tarafından donanımsal olarak gerçekleştirilmesinden sonra, araştırmacıların dikkatini hızla çekmiş ve memristörler üzerinde yoğun çalışmalar ve araştırmalar yapılmasına yol açmıştır (Ascoli et al., 2016; Kvatinsky et al., 2013). Bu atılımın başarılmasına rağmen, memristörün bir devre bileşeni olarak pratikte uygulanması gecikmiştir. Bu gecikme nedeniyle, araştırmacılar memristörlerin rolünü yerine getiren devrelerle devre modellemeye odaklanmış ve çok sayıda modelleme süreci yürütmüştür (Biolek & Biolková, 2009).

Memristörün Yapısı

Memristörün yapısı genellikle iki terminalli, pasif, enerji depolamayan, doğrusal olmayan özelliklere sahip küçük boyutlu bir akım-voltaj devre elemanıdır (Liao et al., 2021). HP firması tarafından donanım olarak hayata geçirilen memristörün matematiksel ifadesi de HP laboratuvarında gerçekleştirilmiştir. İlgili matematiksel ifadeler aşağıda gösterilmiştir. Denklem 1 ile akım ve gerilim arasındaki ilişki ifade edilmiştir.

$$V(t) = R_{MEM}(x)i(t) \tag{1}$$

Denklem 2 ile memristör direncinin değişimi gösterilmiştir.

$$R_{MEM}(x) = [R_{ON}(x) + R_{OFF}(1 - x)] \tag{2}$$

Memristörler, depolanmış ve depolanmamış olmak üzere iki bölge ile ifade edilebilir ve depolanmış bölgenin genişliğinin toplam memristör genişliğine oranındaki değişim, x olarak gösterilmekte olup, aşağıdaki Denklem 3 ile temsil edilmiştir (Strukov et al., 2008).

$$\frac{dx}{dt} = \frac{\mu_v R_{ON}}{D^2} i(t) \tag{3}$$

Daha iyi ve sağlıklı bir sonuç elde edebilmek için HP tarafından önerilen pencereleme fonksiyonu Denklem 4'te ifade edilmiştir.

$$f(x) = \frac{x(1 - x)}{D} \tag{4}$$

Katkılı bölgenin alanının tüm memristör bölgesine oranındaki değişimi bulmak için denklem (4) ve Denklem (3) kullanılarak bu oran bulunur. Bu işlem Denklem 5 ile ifade edilir.

$$\frac{dx}{dt} = \frac{\mu_v R_{ON}}{D^2} i(t) f(x) \quad (5)$$

Memristörler çok küçüktür ve birim alan başına yoğunlukları insan beynindeki sinaps yoğunluğuyla karşılaştırılabilir (Strukov et al., 2008). Bu, biyolojik sinir ağlarının işleyiş yöntemlerini taklit edebilen yapay sinir ağları oluşturmak için umut verici bir teknolojidir (Eshraghian et al., 2022). Bu alanların etkinleştirilebilmesi için memristörlerin büyük diziler halinde düzenlenmesi gerekmektedir (Caravelli & Carbajal, 2018). Memristör dizilerini okuyabilen ve yazabilen devrelerin geliştirilmesi, memristör teknolojisinin ilerlemesinde önemli bir adımdır (Yakopcic et al., 2011). Bu tür devreler, memristör dizisine doğru bir şekilde veri okuyup yazabilmenin yanı sıra memristör dizisinden geçen akım akışını algılama ve kontrol etme gibi diğer işlevleri de yerine getirecek şekilde tasarlanmalıdır. Memristörlerin okunarak ve yazılarak bellek depolama için nasıl kullanılacağını açıklayan makaleler yazılmıştır (Ho et al., 2011; Niu et al., 2010). Bu devreler, bir memristörün durumunu belirlemek için analog karşılaştırıcılar olarak fonksiyonel amplifikatörler kullanır. Kim ve arkadaşlarının çalışması, memristör direncini önceden belirlenmiş sekiz değer kümesine değiştirme yeteneğine sahip karmaşık okuma ve yazma devreleri önermektedir (H. Kim et al., 2010). Memristör, paralel hesaplama ile ilgili olan vektör matris çarpımı için kullanılabilecek doğrusal, çok seviyeli iletim durumlarına sahip olma potansiyeli nedeniyle büyük ilgi görmektedir (Hu et al., 2016). Bu donanım durumu ile yazılım, CPU ve GPU ile birlikte hızlı bir geliştirme ve dönüştürme sürecinde daha yüksek hızlara ulaşma imkanına sahip olmaktadır. Bu nedenle tasarlanan sistemlerden en iyi verimin alınabilmesi için memristörlerin geliştirilen donanım sistemi zincir yapısı içerisinde doğru konumlandırılması önem arz etmektedir.

Sinaps cihazı

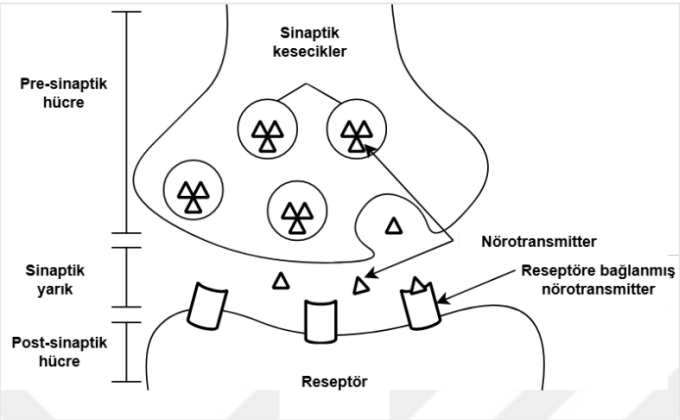
Günümüzde kullanılan hesaplamalı veri işleme, birçok modern bilgisayar sistemi mimarisinin temelini oluşturan Von Neumann mimarisi üzerinde gerçekleştirilmektedir (Tsai et al., 2018). Bu mimari, bir bilgisayarın temel bileşenleri olan işlemci, bellek ve giriş/çıkış birimlerini bir araya getirmektedir (von Neumann, 1993). Merkezi işlem birimi ve belleğin bir arada çalıştığı sistem transistör teknolojisine dayanmaktadır (Shaw, 1950). Bu durum nedeniyle donanım, Moore yasasını takip ederek kısa sürede hızlı bir gelişim göstermiştir (Manfrinato et al., 2013). Moore Yasası'na (Moore, 1965) göre, bir entegre devredeki transistör sayısının her iki yılda bir iki katına çıkacağı ve üretim maliyetlerinin düşmesi nedeniyle bilgisayar donanımlarının performansının da her iki yılda bir iki katına çıkacağı düşünülmektedir. Bu durum daha güçlü ve verimli CPU'ların ve bellek sistemlerinin geliştirilmesine yol açmıştır.

Sonuç olarak, veri işleme donanımı daha yüksek işlem hızları ve daha büyük bellek kapasiteleri sağlayarak daha güçlü ve verimli hale gelmiştir (Hu et al., 2017). Bu durum, daha güçlü ve verimli bilgi işlem sistemlerinin yanı sıra daha gelişmiş yazılım ve uygulamaların geliştirilmesine yol açmıştır. Transistörlerin ölçeklendirilmesi ve daha küçük alanlara daha fazla transistör yerleştirilmesi sonucunda CPU'nun işlem hızı önemli ölçüde artmıştır (Xiu, 2019). Bununla birlikte, CPU ve bellek arasındaki yavaş veri aktarımı, hesaplama hızını yavaşlatır ve bu da enerji verimliliğini azaltır. Bu durum CPU'nun çalışma sırasında daha fazla enerji tüketmesine neden olmaktadır (Zanotti et al., 2020). Moore Yasası sayesinde transistör teknolojisi ve von Neumann mimarisi, transistörleri 10 nanometreye kadar küçülterek sınırlarını zorlamış ve bu teknolojilerin üretim sınırlarına atomik düzeyde ulaşmamızı sağlamıştır (Taur, 2002). CPU'ların ve belleklerin performansını artırmak için birçok araştırmacı transistör ve von Neumann mimarisindeki teknik sorunları ele almaya çalışmıştır (Palit et al., 2014). Bu durum, yüksek performanslı, son teknoloji bilgi işlem ve bellek cihazlarına daha fazla ihtiyaç duyulmasına neden olmuştur. Carver Mead, nöromorfik sistem adı verilen ve aynı anda hem hesaplama hem de bellek işlemlerini gerçekleştirerek insan beyninin çalışma mekanizmasını taklit eden yeni bir hesaplama sistemi olan nöromorfik sistemi önermiştir (Mead, 1990). Nöromorfik sistemler, insan beyninin çalışma mekanizmalarını taklit etmek için geliştirilmiş bir bilgisayar sistemi türüdür. Bunlar, beyin gibi birden fazla işlemi aynı anda gerçekleştirerek çoklu görev yapmak üzere tasarlanmıştır. Nöromorfik sistemler, insan beyninin çalışma mekanizmalarını kopyalamanın yanı sıra, insanların çözebildiği problemleri çözebilme yeteneğine de sahiptir (Hu et al., 2017).

Biyolojik sinaps

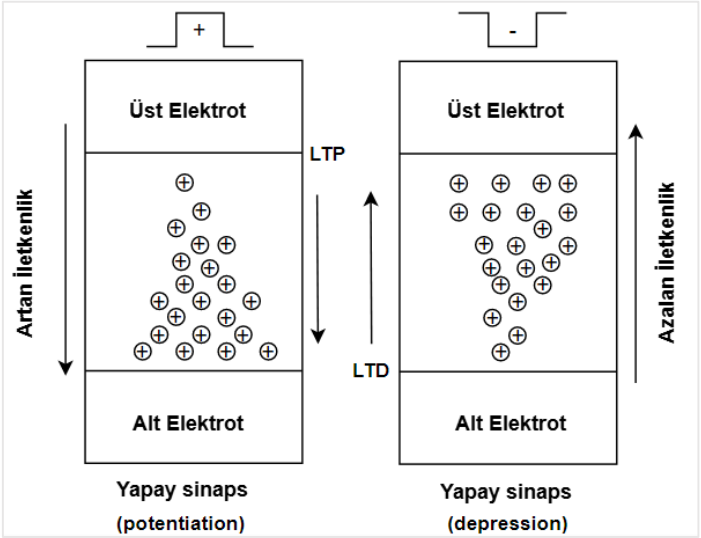
Biyolojik bir sinaps, sinaptik yarık olarak bilinen küçük bir boşlukla birbirine bağlanan iki hücrenin oluşturduğu, sinir hücreleri arasında mesaj iletmek için özel bir bağlantı olarak bulunur (W. Xu et al., 2016). Şekil 6'da sinapsların konumu gösterilmektedir. Şekil 6'da ifade edildiği gibi, pre-sinaptik nöron akson ucundan post-sinaptik nöronun dendritine doğru konumlanır (Kandel et al., 2012). Nörotransmitterler post-sinaptik nöronun reseptörlerine bağlanmadan önce, pre-sinaptik nöronun aksonunun ucunda elektriksel bir spike (darbe) oluştururlar, ardından nörotransmitterler iki nöron arasındaki boşluğa salınır ve postsinaptik nöronun reseptörlerine bağlanarak bir elektrik sinyali oluştururlar (H. Yu et al., 2021). Oluşturulan bu sinyal daha sonra nöron boyunca iletilir. Biyolojik sinyaller, post-sinaptik nöronun dendritine bir nörotransmitter gönderildikten sonra bağlı nöronlar aracılığıyla sinir sistemine iletilir (Bower & Beeman, 1995). Sinaps plastisitesi, sinapsların aktiviteye yanıt

olarak güçlerini değiştirme yeteneğidir. Diğer bir ifadeyle sinapsın bir sinyalin hızını veya yoğunluğunu artırma veya azaltma kabiliyetidir (G. Lee et al., 2021; S. Yu et al., 2011). Ayrıca, kısa ve uzun süreli hafızanın yapılandırılmasıyla yakından bağlantılı olup öğrenme ve hafıza oluşumu için önemlidir (Saïghi et al., 2015).



Şekil 6. Biyolojik sinaps şeması. Sinaptik iletim, presinaptik hücreden gelen sinyallerin postsinaptik hücreye iletilmesi sürecidir. Şemada, sinaptik veziküller presinaptik hücre içinde bulunur ve sinir hücreleri arasında iletişimi sağlayan nörotransmitterleri içerir.

Beynin öğrenme süreci sırasında, sinaps olarak bilinen nöronlar arasındaki bağlantı, sinaptik plastisite aşamasında ya daha güçlü ya da daha depresif hale gelebilir (Rajendran et al., 2019). Spike Zamanına Bağlı Plastisite (STDP), postsinaptik ve pre-sinaptik nöronlardaki aksiyon potansiyellerindeki artışlar arasındaki zamanlama farkına bağlıdır (Burr et al., 2017). Şekil 7, biyolojik bir sinaps ile metal oksit tabanlı yapay bir sinaps arasındaki karşılaştırmayı göstermektedir. Pre-sinaptik ve post-sinaptik nöronlar, uyarıcı ve inhibe edici post-sinaptik potansiyellerin elektrik sinyallerini gönderir ve alır.



Şekil 7. Yapay sinapsın gösterimi.

Bu sinyaller, nörotransmitterleri serbest bırakarak iki nöron arasındaki bağlantının gücünü değiştirmek için kullanılır (Kuzum et al., 2011; Sung et al., 2018). Uzun süreli güçlenme (LTP), sinaptik ağırlığın kalıcı olarak değiştirildiği sinaptik güçte kalıcı bir değişikliktir. Kısa bir süre içinde bir dizi ardışık uyarım meydana geldiğinde, sinaptik aktivite artar (uzun süreli güçlenme) veya azalır (uzun süreli depresyon, LTD). Bu değişiklik birkaç dakika veya daha uzun sürebilir (H. Yu et al., 2021). Spike-Timing-Dependent Plasticity, bir sinapsın gücünün presinaptik ve post-sinaptik aksiyon potansiyellerinin veya spike'ların (STDP) zamanlamasına bağlı olarak değiştiği bir olgudur (Yan et al., 2018).

Yapay sinaps

Biyolojik sinaps plastisitesi, sinaptik ağırlık olarak adlandırılan ve analog bir şekilde kontrol edilebilen bir bağlantı gücüdür (Q. Wan et al., 2019). Yapay sinapslar, biyolojik bir sinapsın önemli işlevlerini taklit etmek için tasarlanmıştır (W. Xu et al., 2016). Yapay sinaps, iki sinir hücresi arasındaki bağlantı olan biyolojik sinapsın işlevini taklit eden bir cihazdır (H. Yu et al., 2021). Yapay sinapslar, insan beyninin yapısını ve davranışını taklit eden bir bilgisayar türü olan nöromorfik bilişimde kullanılmaktadır. Bu sinapslar tipik olarak silikon, metal veya organik moleküller gibi malzemelerden yapılır ve biyolojik sinapslardan geçen elektrik sinyallerini taklit etmek için tasarlanmıştır (H. Yu et al., 2021). Yapay sinapslar, karmaşık sorunları çözmek için kullanılabilecek yapay sinir ağları oluşturmak için kullanılabilir. Şekil 7'de gösterildiği gibi iletkenlik değişiminin (direnç) sinaptik bir ağırlıkla değiştirilmesi, biyolojik nöronun tepkisinin yapay bir sinaps uygulamasıyla taklit edilmesini gerektirir (Do et al., 2010). Yapay sinapslar, hafızanın nasıl korunduğunu belirleyen Uzun Süreli Potansiyasyon (LTP) ve Kısa Süreli Potansiyasyon (STP) olmak üzere iki tür sinaptik plastisiteye sahiptir.

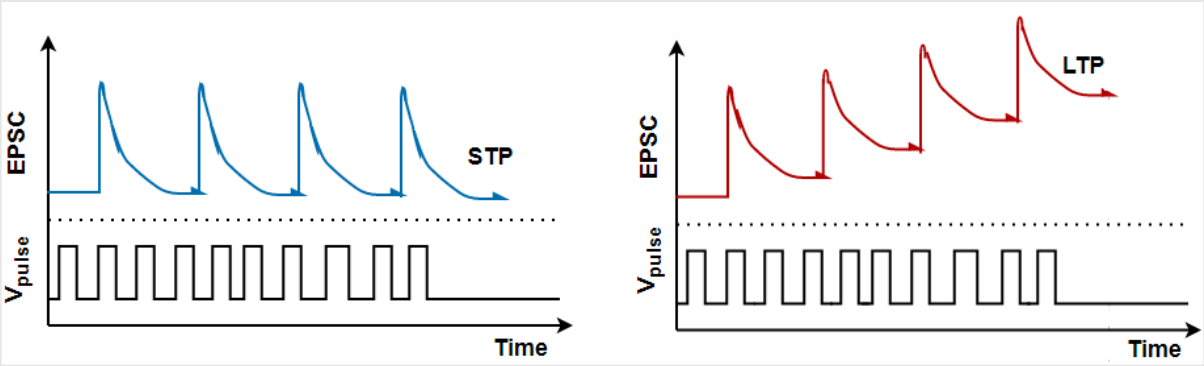
Uzun Vadeli Potansiyasyon (LTP), Uzun Vadeli Depresyon ve Kısa Vadeli Potansiyasyon (LTD)

Sinaptik plastisite, nöronlar arasındaki bağlantıların yoğunluğundaki değişimi ifade eder (Bear & Malenka, 1994). İlk çalışmalarda, yüksek frekanslı presinaptik uyarımın uzun süreli güçlenmeyi (LTP) tetiklediği, düşük frekanslı uyarımın ise sinir hücrelerinde uzun süreli depresyonu (LTD) başlattığı gösterilmiştir (Bliss & Lømo, 1973). Temel faktörlerden biri uzun vadeli plastisite olarak adlandırılır. Sinaptik ağırlıktaki uzun vadeli değişiklikleri ifade eder ve insan beynindeki öğrenme ve hafıza mekanizmalarıyla ilişkili olduğuna inanılır (Daoudal &

Debanne, 2003). Sinirsel bağlantıların güçlendiği, sinaptik güçte uzun vadeli veya kalıcı bir artışla sonuçlanan kalıcı bir duruma uzun vadeli güçlenme (LTP) denir (Feldman, 2012). Diğer bir deyişle, nöral bağlantıların zayıfladığı, sinaptik güçte uzun süreli veya kalıcı bir azalmaya yol açan kalıcı bir durum, uzun süreli depresyon (LTD) olarak adlandırılır (Bear & Malenka, 1994).

LTP ve LTD, nöromorfik bilgi işlem sistemlerinde sinaptik ağırlıkları düzenleyen ve güncelleyen temel ilkeler olarak kullanılabilir (Zeng et al., 2023). Sinaptik plastisite, iki nöron arasındaki bağlantının gücünün değiştirilmesi anlamına gelir ve bu da aralarında iletilen ve sivri uç olarak bilinen darbelerin süresiyle belirlenir (Ranjan et al., 2016; Zahari et al., 2015). Darbe genişliğine dayalı sinaptik plastisite, Spike-Time Dependent Plasticity (STDP) (Dan & Poo, 2006; Linares-Barranco & Serrano-Gotarredona, 2009) olarak bilinen ve sinirsel öğrenme için temel bir mekanizma olarak hizmet eder (Caporale & Dan, 2008).

Elektrokimyasal transistörler, STP veya LTP gibi biyolojik sinapsların nörokimyasal işlevlerini çoğaltmanın bir yolu olarak incelenmekte ve geliştirilmektedir (Sheliakina et al., 2018). STP ve LTP'li sinapslar, plastisite zaman ölçeğinde öğrenme ve hafıza oluşumu için sinirsel temeldir (H. Wang et al., 2018). Çeşitli sistemler LTP süreçlerini mümkün kılarken, sıvı elektrolit bileşenleri (Gkoupidenis et al., 2015; J. Shi et al., 2013) sınırlı tutma süresi ve küçük iletkenlik değişimi nedeniyle belirsiz STP/LTP sinyalleri içerir (Van De Burgt et al., 2017). Çoklu plastisiteye sahip entegre cihazlar oluşturmak ve nöromorfik devreleri basitleştirmek için, güvenilir uçucu olmayan belleğe ve açıkça ayırt edilen kısa vadeli ve uzun vadeli plastisite modlarına sahip katı hal organik sinapslar tasarlamak çok önemlidir. LTP ve STP, Şekil 8'de gösterildiği gibi yapay sinapslar, dış sinyallerle geçici olmayan bir değişime sahip olduklarında gerçekleştirilir.



Şekil 8. Yapay sinapslar LTP (kırmızı) ve STP (mavi).

Makine öğrenimi ile memristör tabanlı çalışmalar

Günümüz dünyasında makine öğrenimi, veri işleme teknolojilerinin temel bileşenlerinden biri haline gelmiştir. Sürekli artan veri hacmiyle birlikte, akıllı veri analizinin teknolojik amaçlar için daha yaygın ve önemli hale geleceğine inanılmaktadır (Mahdavinejad et al., 2018; Sarker et al., 2021). Makine öğreniminin önemli bir kısmı, veri işleme operasyonları ve bu operasyonlar sırasında karşılaşılan sorunlar için etkili çözümler sunmaktır. Makine öğrenimi, yapay zekânın bir alt kümesidir. Makine öğrenimi algoritmaları, açıkça programlanmadan otomatik olarak deneyimlerden öğrenir ve gelişir (Gupta et al., 2022). Makine öğrenimi algoritmaları, verileri akıllıca analiz ederek, gerçek dünya sorunları için akıllı, gerçek zamanlı mühendislik uygulamaları geliştirmenin anahtarıdır (Jhaveri et al., 2022). Makine öğrenimi, Denetimli, Denetimsiz ve Takviyeli Öğrenme olarak kategorize edilir. Memristör tabanlı çalışmalarda (Serb et al., 2016; Z. Wang et al., 2019; Yao et al., 2020; W. Zhang et al., 2023), bu öğrenme algoritmaları çeşitli derin öğrenme yöntemleriyle birlikte kullanılmaktadır.

Makine öğreniminde denetimli öğrenmenin işlevi, her bir girdi öğesini ilgili sınıf etiketi değeriyle eşleştirmektir. Eğitimden sonra, bir bilgisayar bir nesneyi amaçlanan çıktı ile ilişkilendirir, böylece öğrenme sürecini kolaylaştırır. Memristör tabanlı nöromorfik hesaplama sistemleri, sinir ağlarını eğitmek için hızlı ve enerji açısından verimli bir yaklaşım sağlar. Yao ve meslektaşları (Yao et al., 2020) çalışmalarında, bir derin öğrenme üzerinde denetimli bir öğrenme algoritması kullanarak hız ve enerji açısından verimli durumu açıklamışlardır.

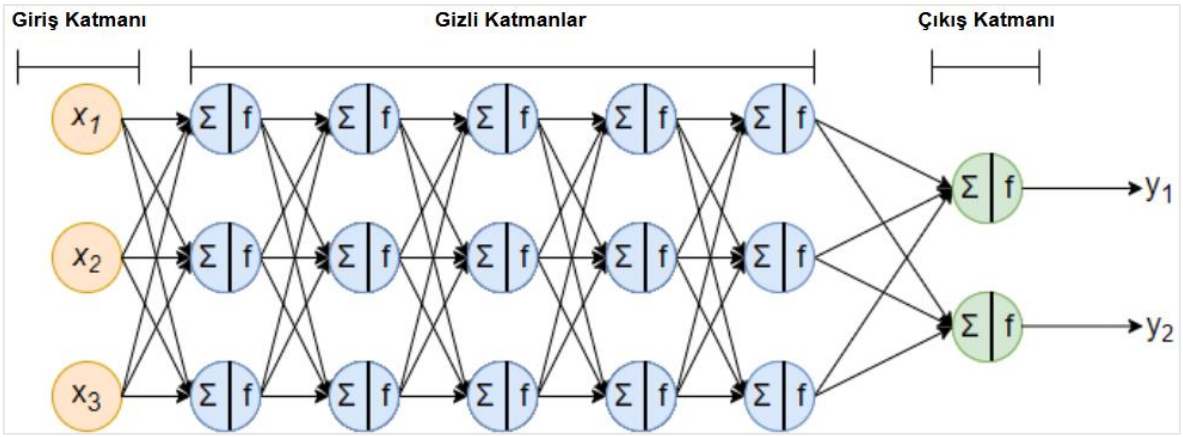
Denetimsiz öğrenme, etiketlenmemiş veri nesneleri üzerinde çalışır. Bu öğrenme türü genellikle özellik çıkarma, önemli desen ve yapıları tespit etme, ilgili nesneleri eşleştirme ve pratik amaçlar için kullanılır (J. Yang et al., 2019). Denetimsiz öğrenme algoritmasının kullanıldığı bir çalışmada, metal oksit tabanlı memristörlerin kademeli ve çok seviyeli anahtarlama özelliklerinden yararlanılarak olasılıksal bir sinir ağı tasarlanmıştır (Serb et al., 2016).

Derin sinir ağı tabanlı pekiştirmeli öğrenme algoritmaları, insan müdahalesi veya gözetimi ihtiyacını azaltarak bilgi ve problem çözme becerilerini otonom olarak edinebilen makineler oluşturma konusunda büyük umut vaat etmektedir (Z. Wang et al., 2019). Wang ve araştırma ekibi çalışmalarında, pekiştirmeli öğrenme uygulamasını sergilemek için deneysel bir çalışma gerçekleştirmiştir. Analog ve dijital bileşenleri birleştiren özel tasarlanmış bir platformda tek bir transistör ve tek bir memristörden (genellikle 1T1R olarak adlandırılır)

oluşan üç katmanlı bir ağ kullandılar. Bunu başarmak için, benzersiz kurulumları için uyarlanmış değiştirilmiş bir öğrenme algoritması kullanmışlardır.

Derin Öğrenmeye Genel Bakış

Derin Öğrenme (Deng, 2014), çok katmanlı sinir ağı yapıları kullanılarak verilerdeki üst düzey soyutlamaları modellemek için çeşitli algoritmalar kullanan bir türdür. İlk olarak 1986 yılında tanıtılmış ve 2000 yılında sinir ağlarına uygulanmıştır (Schmidhuber, 2015). Derin öğrenme algoritmaları, farklı karmaşıklık seviyelerinde veri kümelerini çıkarıp analiz etmek için birden fazla katman kullanır (LeCun et al., 2015). Derin öğrenme uygulamaları, bilgisayarların basit kavramlardan karmaşık kavramları öğrenip analiz etmelerine olanak tanır (Goodfellow et al., 2016). Hiyerarşik öğrenme olarak da bilinen Derin Öğrenme (DL), Yapay Sinir Ağı'nın (ANN) genel aktivasyonunu dönüştürmek için birçok hesaplama aşamasında ilgili değerlerin doğru şekilde atanmasıyla ilişkilidir (Deng, 2014). Derin mimariler, çok düzeyli soyutlamalarla (yani doğrusal olmayan işlemlerle) karmaşık fonksiyonları öğrenme işleminde kullanılır (Turian et al., 2009). Kısacası, derin öğrenme, denetimli veya denetimsiz; özellik öğrenme, sınıflandırma ve desen tanıma gibi görevleri yerine getirmek için çok düzeyli doğrusal olmayan hesaplama ve soyutlama kullanan makine öğreniminin bir dalıdır (Deng, 2014).



Şekil 9. Derin Sinir Ağı

Yapay Sinir Ağı (ANN), katmanlar halinde düzenlenmiş yapay nöronlardan oluşan bir hesaplama sistemidir. Derin Sinir Ağları (DNN'ler) günümüzde bilgisayarla görme, konuşma tanıma ve robotik gibi birçok yapay zekâ ve makine öğrenimi uygulaması için yaygın olarak kullanılmaktadır (Sze et al., 2017). Bu sinir ağlarının klasik uygulamalarda gerçekleştirdiği hesaplama işlemleri, bellek ve işlemci birimleri arasında büyük ve sürekli veri hareketleri gerektirir; bu tür iş yükleri için özel donanım geliştirme çalışmaları devam etmektedir (Musisi-

Nkambwe et al., 2021). Şekil 9'da gösterilen ağ yapısı, giriş verilerini tanımlamak ve sınıflandırmak için birlikte çalışan, her biri farklı ağırlık değerlerine sahip, birbiriyle bağlantılı düğüm katmanlarından oluşmaktadır. Şekil 9'da gösterilen daireler ağın düğümleri olarak adlandırılır. Her düğüm belirli girdi değerlerine sahiptir ve onları birbirine bağlayan çizgilere ağırlık denir.

Derin Sinir Ağları (DNNs)

Derin Sinir Ağları (DNN'ler) (Oli-Uz-Zaman et al., 2022; J. Zhang & Zong, 2015), geleneksel algoritmalarla çözülemeyen karmaşık problemleri çözmek için kullanılır. Bu ağlar, verilerden karmaşık desenleri ve özellikleri çıkarma yeteneğine sahiptir. DNN'ler, görüntü tanıma ve doğal dil işleme gibi çeşitli görevlerde de kullanılabilir. DNN, veriyi ileri iletmek için çok katmanlı doğrusal olmayan gizli düğümler kullanan bir yapay sinir ağıdır (Capra et al., 2020). DNN'ler üzerine yapılan araştırmalar, derin katmanlı ağlar, filtreler, eğitim ve test veri setlerini içerir. Yapay sinir ağları, katman sayısının artırılmasıyla daha derin hale getirilerek derin sinir ağları (DNN'ler) oluşturulur. Araştırmacılar, çalışmalarında derin inanç ağları (DBN), derin yığın ağları (DSN), evrişimli sinir ağları (CNN) ve yinelemeli sinir ağları (RNN) gibi çeşitli DNN sistemleri geliştirmişlerdir (J. Zhang & Zong, 2015).

Tablo 1. DNN tipleri (Cheng & An, 2021)

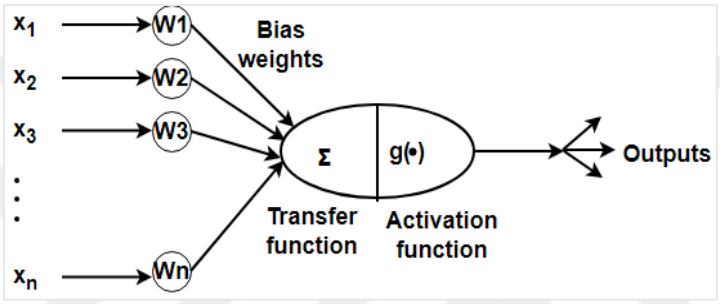
DNNs	Temel özellikler	Temel uygulamalar
CNN	Evrişim, havuzlama ve tam bağlantı katmanlarının etkisinin azalması.	Görüntü İşleme ve Nesne Tespiti
RNN	Yapay sinir ağı yapısında ileri beslemeli ve geri beslemeli sinir ağı yapıların kombinasyonu (geri besleme).	Doğal Dil İşleme, Kredi Kartı Dolandırıcılığı Tespiti, El Yazısı Tanıma.
LTSM	Özel bir RNN türü olan derin öğrenme algoritmalarıyla, görüntülerden elde edilen bilgiler kullanılarak birçok sınıflandırma ve tahmin işlemi gerçekleştirilebilir.	Doğal dil işleme ve metin işleme: Görsellerden otomatik altyazı oluşturma ve ilgili metinlerden kelime üretme.
Gan	Denetimsiz öğrenme, Generatif Model ve Ayırt Edici Model olmak üzere iki modeli içerir.	Anlamsal Segmentasyon ve Görüntü Çözünürlüğü Artırma.

Derin sinir ağları (DNN'ler), görüntü sınıflandırma veya konuşma tanıma gibi karmaşık makine öğrenimi görevlerinde olağanüstü sonuçlar göstermiştir. DNN makine öğrenimi teknikleri, birçok uygulamada insan performansına eşdeğer veya bazı durumlarda daha yüksek doğruluk seviyelerine ulaşmıştır. Uygulamalarda, belirli bir görev için elde edilen yüksek doğruluğun, verideki herhangi bir yapaylıktan değil, uygun problem temsilinin kullanılmasından kaynaklandığından emin olmak önemlidir (Leek et al., 2010; Soneson et al.,

2014). Bu nedenle, modelin öğrendiklerini yorumlamak ve anlamak için geliştirilen yöntemler, güvenilir bir doğrulama sürecinin önemli bir parçası haline gelmiştir (Hansen et al., 2011). Yorumlanabilirlik, özellikle tıp veya otonom araçlar gibi modelin doğru özelliklere bağlı kalarak çalışmasının hayati önem taşıdığı uygulamalarda çok büyük öneme sahiptir (Bojarski et al., 2017; Caruana et al., 2015). Tablo 1'de DNN'ler kendine özgü özellikleri ve uygulama alanları ile gösterilmiştir.

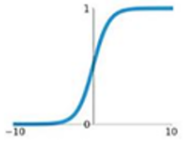
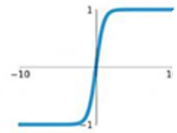
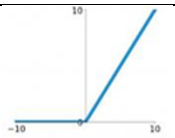
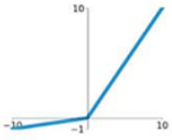
DNN'ler ve aktivasyon fonksiyonları

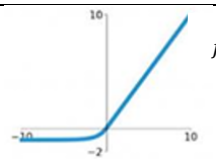
Literatürde, sigmoid, lojistik, tanh, ReLU ve Leaky-ReLU gibi aktivasyon fonksiyonları bulunmaktadır (Hoon Chung et al., 2016).



Şekil 10. Aktivasyon fonksiyonunun basitleştirilmiş blok diyagramı.

Tablo 2. Aktivasyon fonksiyonları ve özellikleri

Aktivasyon Fonksiyonu	Grafik		Tanım
Sigmoid		$\sigma(x) = \frac{1}{1 + e^{-x}}$	Sigmoid aktivasyon fonksiyonu, sıfır ile bir arasında bir değer üretir.
Tanh		$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	Tanh fonksiyonu, -1 ile +1 arasında değerler üretir.
ReLU		$\max(0, x)$	ReLU aktivasyon fonksiyonunda, giriş değeri sıfırdan küçük olduğunda çıkış değeri sıfırdır, ancak giriş değeri sıfırdan büyük olduğunda çıkış değeri giriş değeriyle eşittir (Nwankpa et al., 2018).
Leakly ReLU		$\max(0.1x, x)$	Leaky ReLU fonksiyonu, klasik ReLU aktivasyon fonksiyonunun bir varyasyonudur. Bu fonksiyonun çıkışı, negatif giriş değerlerine karşı küçük bir eğime sahiptir (J. Xu et al., 2020).

ELU		$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$	Negatif değerlere sahip olması, ortalama birim aktivasyonunu sıfıra yaklaştırarak hesaplama karmaşıklığını azaltır ve bu sayede öğrenme hızını artırır (J. Xu et al., 2020).

Aktivasyon fonksiyonunun rolü, belirli bir nöronun aktive edilip edilmeyeceğine karar vermek gibi çeşitli aktivasyon fonksiyonları vardır (Sharma et al., 2020). Bu, bir katmandaki nöronlardan gelen çıktı değerlerinin aktivasyon fonksiyonları kullanılarak sonraki katmanlardaki nöronlara aktarıldığı anlamına gelir. Nöronda oluşan bir çıktının diğer katmanlara gönderilip gönderilmeyeceğine karar verilirken bir eşik değerine ihtiyaç duyulur. Bunun nedeni nörondaki verinin değerinin (-∞, +∞) aralığında herhangi bir değer alabilmesidir. Ayrıca nöron veri transfer limitini bilmemektedir. Bu nedenle, nöron aktivasyonunda karar verme sürecinde aktivasyon fonksiyonları kullanılmalıdır. Bir aktivasyon fonksiyonu kullanılmadan, ağırlıklar ve bias değerleri yalnızca doğrusal bir dönüşüm gerçekleştirecektir, bu da doğrusal bir regresyon modeliyle aynıdır (Jagtap et al., 2020). Bu doğrusal modelin çözülmesi kolaydır, ancak karmaşık problemlerle çok iyi sonuçlar başaramaz. Doğrusal olmayan aktivasyon fonksiyonu, giriş verileri üzerinde doğrusal olmayan bir dönüşüm gerçekleştirerek daha karmaşık görevleri öğrenmesine ve gerçekleştirmesine olanak tanır (Jagtap et al., 2020). Basitleştirilmiş aktivasyon fonksiyonu yapısı Şekil 10'da gösterilmektedir (Haoxiang & S, 2021).

Tablo 2, yapay sinir ağlarında kullanılan çeşitli aktivasyon fonksiyonlarının grafiksel gösterimleri verilmiştir.

DNN katman tipleri

Derin öğrenme, yapay zekâ disiplininde, bilgisayarlı görü, doğal dil işleme ve konuşma tanıma gibi alanlarda çığır açıcı gelişmelere imkân tanıyan bir makine öğrenimi alt dalıdır. Bu alandaki başarılar, büyük ölçüde çok katmanlı yapay sinir ağları adı verilen karmaşık hesaplamalı modellerin geliştirilmesine dayanmaktadır. Her bir sinir ağı katmanı, belirli bir özelliği öğrenmeye yönelik olarak tasarlanmış olup, tüm katmanların bir araya gelmesiyle oluşturulan derin mimari, büyük veri kümelerinden karmaşık ilişkileri çıkarabilme yeteneği kazanır. Cai, Wu ve Zhang tarafından yapılan çalışmada vurgulandığı üzere, çalışmada incelenen derin öğrenme mimarilerinde kullanılan farklı katman türleri, ağın genel performansını doğrudan etkileyen optimizasyon süreçlerinin merkezinde yer almaktadır (Y. Cai et al., 2019; Y.-H. Jin et al., 2020). Bu bağlamda, derin öğrenme modellerinde sıklıkla

kullanılan yinelemeli katmanlar, havuzlama katmanları, tam bağlantılı katmanlar ve döngüsel katmanlar gibi farklı katman türlerinin özellikleri ve işlevleri kullanılmaktadır. Bu tezin kapsamında uygulamada kullanılan katmanlar ve hipermatreler ayrıntılı olarak ilerleyen bölümlerde ele alınacaktır.

Giriş katmanı

Giriş katmanı, derin öğrenme modellerinde ham verilerin, modelin iç yapısına entegre edildiği ilk ve kritik bir bileşendir. Bu katman, dış dünyadan elde edilen karmaşık ve çeşitli veri türlerini, modelin anlayabileceği bir temsile dönüştürme işlevini üstlenir. Bilgisayarlı görü uygulamalarında olduğu gibi, giriş katmanı, ham görüntü verilerini standartlaştırmak, gürültüleri azaltmak ve belirgin özellikleri vurgulamak amacıyla ön işleme adımları gerçekleştirir. Örneğin, Feng ve arkadaşları tarafından yapılan çalışma (Feng et al., 2020) ile giriş katmanında uygulanan çeşitli görüntü dönüşümlerinin, modelin öznitelik öğrenme performansını önemli ölçüde etkilediği gösterilmiştir. Bu sayede, model, daha yüksek seviyeli soyut kavramları öğrenerek, karmaşık görsel verileri daha doğru bir şekilde sınıflandırabilir veya nesneleri tespit edebilir. Dolayısıyla, giriş katmanı, derin öğrenme modellerinin başarısı için temel bir yapı taşıdır ve modelin genel performansını doğrudan etkileyen bir faktördür.

RNN katmanı

Tekrarlayan sinir ağları (RNN), özellikle sıralı veriyle çalışan derin öğrenme modellerinin temel yapı taşlarından biridir. Bu ağlar, giriş verilerindeki zaman veya sıraya bağlı bağlamları anlamak ve işlemek amacıyla tasarlanmış yapısal bileşenlerdir. RNN'ler, her bir zaman adımında girdileri işleyerek ve önceki zaman adımından gelen bilgileri bir "gizli durum" aracılığıyla koruyarak, verilerdeki ardışık ilişkileri öğrenir. Bu mekanizma, metin, ses ve zaman serileri gibi sıralı verilerde önemli olan bağlamsal bilgilerin modellenmesini sağlar. RNN'in temel çalışma prensibi, bir ağırlık matrisi aracılığıyla giriş vektörleri ve önceki gizli durumun birleşik bir dönüşümünü gerçekleştirmek ve bu dönüşüme doğrusal olmayan bir aktivasyon fonksiyonu uygulamaktır. Bu süreç, her bir zaman adımında tekrarlanarak verideki uzun vadeli ve kısa vadeli bağımlılıkların yakalanmasını mümkün kılar. Ancak, RNN'lerin vanishing gradient (kaybolan gradyan) problemi nedeniyle uzun vadeli bağımlılıkları öğrenmekte zorluk yaşayabileceği bilinmektedir. Bu sorunu çözmek için, LSTM (Uzun Kısa Süreli Bellek) ve GRU (Gated Recurrent Unit) gibi gelişmiş RNN türleri geliştirilmiştir. Bu türler, veri içinde hangi bilgilerin tutulacağı veya unutulacağına karar veren kapı mekanizmalarını kullanarak performansı artırır (Hochreiter & Schmidhuber, 1997). Tekrarlayan sinir ağları, derin öğrenme

modellerinin sıralı verileri anlamasında ve sıralı veri gerektiren görevlerde yüksek performans göstermesinde kritik bir rol oynar. RNN'ler, bu bağlamsal ilişkileri yakalayarak makine çevirisi, duygu analizi ve ses tanıma gibi uygulamalarda derin öğrenmenin başarısını artırmaktadır.

Havuzlama

Havuzlama katmanları, derin öğrenme tabanlı bilgisayarlı görme sistemlerinde, yinelemeli katmanlarla birlikte kullanılan temel bir bileşendir. Bu katmanlar, RNN işlemi sonucu elde edilen öznitelik haritalarının boyutunu düşürmek amacıyla alt örnekleme işlemi uygularlar. Alt örnekleme, hesaplama maliyetini azaltarak modelin daha verimli çalışmasını sağlar, aşırı öğrenme riskini düşürür ve çeviri değişmezliği artırarak modelin farklı boyutlardaki veya hafifçe deforme olmuş girdilere karşı daha dayanıklı olmasını sağlar. Yaygın olarak kullanılan havuzlama yöntemleri arasında, bir bölge içerisindeki en büyük değerin seçildiği maksimum havuzlama ve bölge içerisindeki değerlerin ortalamasının alındığı ortalama havuzlama bulunmaktadır. Yapılan bir çalışma (S. Li et al., 2019) ile havuzlama katmanlarının derin öğrenme modellerinin başarısı üzerindeki etkileri detaylı bir şekilde incelenmiştir . Bu çalışmada, farklı havuzlama yöntemlerinin modelin performansı, öğrenme süresi ve genelleme yeteneği üzerindeki etkileri karşılaştırılmıştır.

Tam bağlantılı katman

Tam bağlantılı katmanlar (yoğun katmanlar olarak da bilinir), derin öğrenme modellerinde, verideki karmaşık ilişkileri ve soyut kavramları öğrenmek için kullanılan temel bir bileşendir. Bu katmanlarda, bir önceki katmandaki her nöron, sonraki katmandaki her nörona tek tek bağlanır. Bu sayede, ağ boyunca bilgi akışı maksimum düzeyde sağlanır ve model, verideki global bilgiyi daha etkili bir şekilde yakalar. Tam bağlantılı katmanlar, genellikle derin öğrenme modellerinin çıkış katmanı olarak kullanılır ve sınıflandırma veya regresyon gibi çeşitli görevlerde son tahmini yaparlar. Sınıflandırma görevlerinde, softmax gibi bir aktivasyon fonksiyonu kullanarak farklı sınıflar için olasılık dağılımları elde edilirken, regresyon görevlerinde ise doğrusal bir aktivasyon fonksiyonu tercih edilir. Yapılan bir çok çalışma (Prihatno et al., 2021; Q. Wang et al., 2020) ile tam bağlantılı katmanların derin öğrenme modellerindeki önemini ve çeşitli uygulamalarını detaylı bir şekilde incelemiştir. Bu çalışmalarda, tam bağlantılı katmanların farklı mimarilerdeki ve hiperparametre ayarlarındaki etkileri incelenerek, model performansını optimize etmek için önemli bulgular elde edilmiştir.

Bırakma katmanı

Bırakma katmanı, derin öğrenme modellerinde aşırı uyum sorununu hafifletmek için sıklıkla kullanılan bir düzenleme tekniğidir. Aşırı uyum, modelin eğitim verilerine aşırı derecede uyum sağlaması ve sonuç olarak yeni, görülmemiş veriler üzerinde genelleme yeteneğini kaybetmesi durumudur. Bırakma tekniği, bu sorunu çözmek amacıyla, eğitim sürecinde sinir ağının gizli katmanlarındaki nöronların bir kısmını rastgele devre dışı bırakarak modelin karmaşıklığını azaltır. Bu sayede model, eğitim verilerindeki rastlantısal gürültülere karşı daha dayanıklı hale gelir ve genelleme performansı artar. Yapılan bir çalışma (Choe & Shim, 2019) ile bırakma tekniğinin, özellikle büyük ve karmaşık sinir ağlarında aşırı uyumu önlemedeki etkinliği vurgulanmıştır. Bırakma işlemi, her bir eğitim yinelemesinde farklı bir alt kümedeki nöronların devre dışı bırakılması şeklinde gerçekleştirilir. Bu sayede, model, her bir eğitim adımında farklı bir yapıya sahip olur ve bu da modelin daha genelleyici özelliklere sahip olmasını sağlar. Sonuç olarak, bırakma katmanı, derin öğrenme modellerinin başarısı için önemli bir araçtır ve modelin genelleme yeteneğini önemli ölçüde artırır.

Düzleştirme katmanı

Düzleştirme katmanı, derin öğrenme mimarilerinde, özellikle evrişimli sinir ağlarında, çok boyutlu girdi verilerini tek boyutlu bir vektör temsiline dönüştürmek amacıyla kullanılan bir işlemdir. Bu işlem, yüksek boyutlu verilerin, tamamen bağlantılı katmanlar gibi daha basit yapıdaki katmanlara uygulanabilmesi için gereklidir. Evrişimli katmanlar tarafından elde edilen özellik haritaları gibi çok boyutlu veriler, düzleştirme işlemi sayesinde tek boyutlu bir vektör haline getirilerek, sonraki katmanlara daha uygun bir formatta sunulur. Bu sayede, modelin daha yüksek seviyedeki soyut kavramları öğrenmesi ve karmaşık kararlar alması kolaylaşır. Anand ve arkadaşları tarafından yapılan çalışmada, düzleştirme katmanının, derin öğrenme modellerinin başarısı üzerindeki etkisi ve farklı mimarilerdeki kullanımı detaylı bir şekilde incelenmiştir (Anand et al., 2022). Düzleştirme işlemi, herhangi bir parametre öğrenimi gerektirmeyen basit bir işlem olmasına rağmen, modelin genel performansı üzerinde önemli bir etkiye sahip olabilir. Özellikle, düzleştirme işlemi öncesinde uygulanan normalizasyon ve beyazlatma gibi ön işleme teknikleri, modelin öğrenme hızını artırabilir ve genelleme yeteneğini güçlendirebilir.

DNN modellerinin hiperparametreleri ve uyarlanmaları

Derin öğrenme modellerinin tasarımı, belirli bir problem için en uygun sinir ağı mimarisinin belirlenmesi sürecini ifade eder. Bu süreçte, modelin yapısal bileşenleri olan

katman sayısı, katman türleri ve bu katmanların birbirine nasıl bağlanacağı gibi kritik kararlar verilir. Modelin performansını doğrudan etkileyen hiperparametreler ise, modelin öğrenme sürecini kontrol eden ayarlanabilir parametrelerdir. Öğrenme oranı, küme boyutları, düzenleme parametreleri ve optimizasyon algoritması gibi hiperparametreler, modelin genelleme yeteneği, eğitim süresi ve yakınsama hızı üzerinde önemli etkilere sahiptir. Bu parametrelerin uygun şekilde ayarlanması, modelin istenen performansa ulaşabilmesi için büyük önem taşır. Derin öğrenme modellerinin tasarımında hem teorik bilgi hem de deneysel çalışmaların birleşimiyle elde edilen bilgilerden yararlanılır. Modelin karmaşıklığı, veri kümesinin özellikleri, hesaplama kaynakları ve çözülmek istenen problemin doğası gibi faktörler, tasarım sürecinde dikkate alınması gereken önemli unsurlardır. Tasarlanan bir DNN modelinde birçok hiperparametre bulunabilir. Bu parametrelerden en önemlileri; hedef boyutu, parti boyutu, aktivasyon fonksiyonu, optimizasyon fonksiyonu, eğitim tur sayısı olarak sıralanabilir.

Bir derin öğrenme modelinin başarısı, sadece mimarisiyle değil, aynı zamanda hiperparametrelerinin de doğru seçimiyle belirlenir. Hiperparametreler, modelin öğrenme sürecini kontrol eden ve modelin genelleme yeteneğini doğrudan etkileyen ayarlanabilir değerlerdir. Bu nedenle, en uygun hiperparametreleri belirlemek, model tasarımının kritik bir aşamasıdır. Hiperparametre optimizasyonu, genellikle deneysel bir süreçtir ve araştırmacıların domain bilgisine, sezgisine ve mevcut yöntemlere dayanır. Son zamanlarda, bu süreci daha verimli hale getirmek için otomatik hiperparametre arama yöntemleri popülerlik kazanmaktadır. Aşağıda, derin öğrenme modellerinde sıkça kullanılan hiperparametreler ve bunların etkileri detaylı olarak incelenecektir.

Hedef boyut

Hedef boyut parametresi, görüntü işleme görevlerinde, girdi görüntülerinin çözünürlüğünü düşürerek hesaplama maliyetini azaltmayı amaçlayan bir tekniktir. Bu yöntem, orijinal görüntüde bazı bilgilerin kaybına yol açsa da modelin eğitim ve tahmin süreçlerini hızlandırır. Ancak, aşırı derecede çözünürlük düşürmek, modelin önemli görsel özelliklerini kaçırmasına ve performansını düşürmesine neden olabilir. Bu nedenle, hedef boyut parametresinin seçimi, modelin doğruluğu ve hızının bir denge noktası olarak belirlenmelidir.

Batch boyutu

Derin öğrenme algoritmalarında, tüm eğitim verisi üzerindeki hesaplamaların tek seferde gerçekleştirilmesi, özellikle büyük veri setleri için hesaplama maliyetini önemli ölçüde artırabilir. Bu nedenle, eğitim verisi daha küçük alt kümelere bölünerek işlenir. Bu alt kümelere

"parti" veya "mini-batch" denir. Parti boyutu ise, her bir partide bulunan örnek sayısını belirten bir hiperparametredir. Parti boyutunun seçimi, modelin yakınsama hızını, genelleme performansını ve bellek kullanımını etkileyen önemli bir faktördür. Küçük parti boyutları, modelin daha sık güncellenmesini sağlayarak daha hızlı öğrenmeye olanak tanırken, büyük parti boyutları ise daha kararlı bir öğrenme süreci sunar.

Aktivasyon fonksiyonu

Derin öğrenme modellerinde, aktivasyon fonksiyonları, doğrusal bir dönüşümün ardından elde edilen çıktıyı doğrusal olmayan bir çıktıya dönüştürerek modelin karmaşık ilişkileri öğrenme yeteneğini artırır. Bu sayede model, gerçek dünyadaki verilerin doğrusal olmayan yapısını daha iyi temsil edebilir. Aktivasyon fonksiyonları, aynı zamanda, geri yayılım algoritması ile modelin parametrelerinin güncellenmesi için gerekli olan gradyan hesaplamalarında da önemli bir rol oynar. Sigmoid, tanh ve ReLU gibi farklı aktivasyon fonksiyonları bulunmaktadır. ReLU, genellikle tercih edilen bir fonksiyon olup, daha hızlı öğrenme ve daha az hesaplama maliyeti gibi avantajlara sahiptir (Gong et al., 2023).

Optimizasyon fonksiyonu

Derin öğrenme modellerinin eğitimi, özünde, yüksek boyutlu bir parametre uzayında bir optimizasyon problemidir (Mai et al., 2023). Modelin parametrelerini, belirli bir maliyet fonksiyonunu minimize edecek şekilde ayarlamak amacıyla çeşitli optimizasyon algoritmaları kullanılır. Bu algoritmalar, genellikle stokastik gradyan inişi (SGD) yöntemine dayalı olup, model parametrelerini, maliyet fonksiyonunun gradyantının zıt yönünde güncelleyerek iteratif bir şekilde ilerler.

Stokastik gradyan inişin temel bir varyasyonu olan SGD, her iterasyonda eğitim verisinin küçük bir alt kümesi (mini-batch) üzerinde hesaplanan gradyant kullanılarak parametreleri günceller. Bu sayede, hesaplama maliyeti azaltılır ve model daha hızlı öğrenir. Ancak, SGD, öğrenme oranının uygun şekilde ayarlanmasını ve yerel minimumlara sıkışma riskini içerir. Bu sorunları aşmak için, Adagrad, Adadelta, Adam, Adamax gibi adaptif öğrenme oranı algoritmaları geliştirilmiştir. Bu algoritmalar, her bir parametre için farklı öğrenme oranları kullanarak, seyrek parametrelerin daha hızlı öğrenmesini ve sık görülen parametrelerin daha yavaş öğrenmesini sağlar.

Farklı optimizasyon algoritmalarının performansı, veri kümesi, model mimarisi ve hiperparametre ayarlarına göre değişebilir. Örneğin, Adagrad, seyrek veri setlerinde iyi

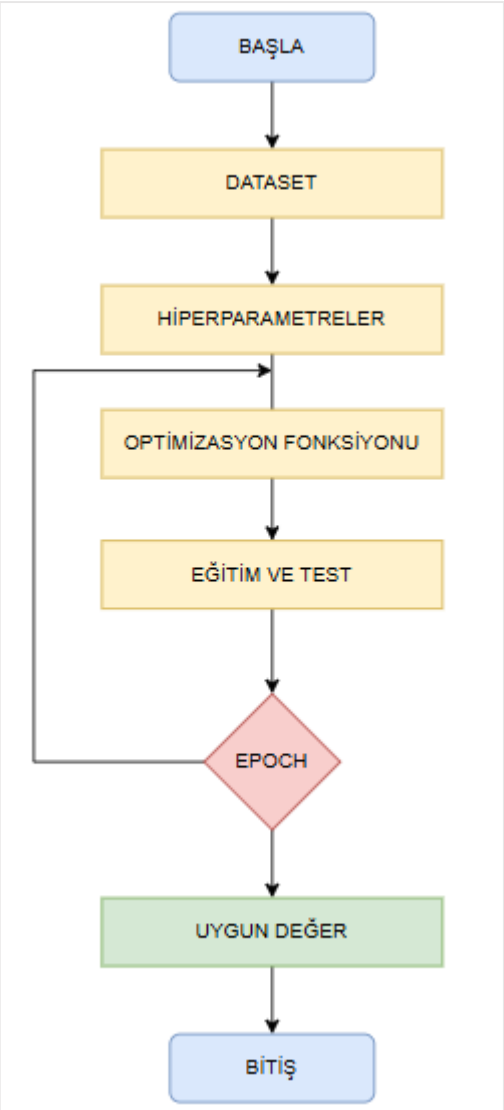
performans gösterirken, Adam, daha genel amaçlı bir optimizasyon algoritması olarak kabul edilir. Adamax ise, Adam'ın bir varyasyonu olup, daha kararlı bir öğrenme süreci sunar. Seçilen optimizasyon algoritması, modelin yakınsama hızını, genelleme performansını ve hesaplama maliyetini doğrudan etkiler. Bu nedenle, derin öğrenme uygulamalarında, farklı optimizasyon algoritmalarını deneyerek en uygun olanını seçmek önemlidir. Optimizasyon yöntemleri bu tez çalışmasının önemli bir parçası olup ileride detaylı biçimde ele alınacaktır.

Epoch sayısı

Eğitim sürecinde, tüm eğitim verisi üzerinde bir geçiş yapılması "epoch" olarak adlandırılır. Her epoch'ta, eğitim verisi belirli büyüklükteki parçalara (batch veya mini-batch) ayrılır ve model bu parçalardaki veriler üzerinde sırayla eğitilir. Her batch'ten sonra, modelin ağırlıkları, hesaplanan gradyantlar doğrultusunda güncellenir. Bu iteratif süreç, modelin parametrelerinin optimum değerlere yakınsamasını sağlar.

Epoch sayısı, modelin ne kadar süre eğitildiğini belirleyen önemli bir hiperparametredir. Yetersiz sayıda epoch ile model tam olarak öğrenmeyebilir ve düşük performans gösterebilir. Ancak, aşırı sayıda epoch ise aşırı öğrenmeye (overfitting) neden olabilir, yani model eğitim verisine aşırı uyum sağlayarak yeni, görülmemiş verilere genelleme yeteneğini kaybedebilir.

Epoch sayısı, derin öğrenme modellerinin eğitiminde kritik bir rol oynar. Hem yetersiz hem de aşırı sayıda epoch, modelin performansını olumsuz etkileyebilir. Bu nedenle, optimum epoch sayısını belirlemek için dikkatli bir hiperparametre ayarlaması yapmak gerekmektedir. Erken durdurma gibi teknikler, bu optimizasyon sürecinde önemli bir araçtır. DNN'deki temel yapışlar Şekil 11'de gösterilmiştir.



Şekil 11. DNN metodolojisinin akış diyagramı.

Memristör tabanlı DNN çalışmaları

Son yıllarda, elektronik, hesaplama ve iletişim teknolojilerindeki hızlı gelişmeler, derin öğrenme sistemlerinin geliştirilmesi sayesinde veri yoğun hesaplamayı daha erişilebilir ve doğru hale getirmiştir. Bu durum, daha büyük, daha enerji verimli ve daha hızlı sinir ağlarının oluşumuna yol açmıştır (Josh & Gibson, 2017). Bu gelişmeler, yeni fiziksel ekipmanların ve bilgisayar teknolojilerinin geliştirilmesini teşvik etmiş ve yapay zekanın karmaşık sorunları çözmek için kullanılabilmesini mümkün kılmıştır (Cheng & An, 2021). Ayrıca, bu teknoloji, verilerdeki desenleri tanımlamak ve tahminler yapmak için kullanılabilecek daha gelişmiş makine öğrenmesi algoritmalarının geliştirilmesini sağlamıştır (Schuman et al., 2017).

Memristör, entegre çiplerde büyük ölçekte sinir ağları oluşturmayı mümkün kılan yeni bir donanım türüdür (Cheng & An, 2021). Geleneksel CMOS lojik devrelere kıyasla, memristör

devrelerini yapay sinir ağı sistemlerini simüle etmek için kullanmak, daha küçük alan, daha düşük enerji tüketimi ve sensör simülasyon hesaplamaları için benzer yetenekler gibi avantajlar sunmaktadır. Ancak, memristörlerin DNN uygulamalarında kullanımına ilişkin bazı zorluklar da bulunmaktadır; donanım değişkenliği, bu alandaki teknolojinin olgunlaşmamış olması ve memristörlerin dayanıklılığı gibi sorunlar yapay sinir ağlarının memristörlerle geliştirilmesini engellemekte ve araştırmacılara aşılması gereken zorluklar sunmaktadır. Literatürde bu problemlerin çözülmesiyle, memristör tabanlı sistemlerin, günümüzde kullanılan von Neumann mimarisine dayalı klasik sistemlere kıyasla enerji ve hız açısından çok daha verimli olacağı belirtilmektedir (Ye et al., 2022). Bu yeni alanın sunduğu avantajlar ve mevcut sistemin tıkanıklığı nedeniyle, araştırmacıların yoğun olarak çalıştığı bir alana dönüşmeye başlamıştır.

Li ve arkadaşları, çalışmalarında memristör tabanlı çok katmanlı sinir ağlarını kullanarak yaklaşık hesaplamalar için enerji verimli bir uygulama gerçekleştirmiştir (B. Li et al., 2013). Yaklaşık hesaplama sürecini hızlandırmak için programlanabilir bir memristöre önce Yaklaşık Hesaplama Birimi (Memristör ACU) öğretilmiştir (B. Li et al., 2013). Ardından, Memristör ACU'nun üzerinde ölçeklenebilir bir memristör tabanlı yaklaşık hesaplama çerçevesi önerilmiştir. Bu sistemde uygulanan işlevlerdeki maksimum hata oranı %1,87 olup, enerji tüketimi benzerlerine göre 22 kat daha verimlidir.

Yuan ve arkadaşları, çalışmalarında (Yuan ve diğerleri, 2019) memristör çapraz çubuk dizisini von Neumann mimarisinin zorluklarını azaltmak ve derin sinir ağlarının (DNN'ler) düşük güç tüketimiyle hızlandırılmasını sağlamak için teşvik edici bir çözüm olarak kullanmışlardır. Memristör tabanlı ağırlık budama ve ağırlık quantizasyonunu ayrı ayrı inceleyerek, orijinal DNN modeline kıyasla alan ve enerji tasarrufu sağladıklarını göstermişlerdir. Yüksek doğruluk, düşük güç ve küçük bir alan elde etmek amacıyla çapraz çubuk bloklarının budanması, iletkenlik aralığı ve ağırlık değerleri ile gerçek cihazlar arasındaki uyumsuzluk gibi donanım sınırlamalarını da göz önünde bulundurmuşlardır. Deneysel sonuçlar, önerilen yöntemin VGG-16 ağında %29,81 oranında ağırlık sıkıştırma ile güç ve alan tüketiminde sırasıyla %98,38 ve %98,29 azalma sağladığını ve orijinal DNN modellerine kıyasla yalnızca %0,5 doğruluk kaybına yol açtığını göstermektedir.

Derin sinir ağlarının popülaritesi, yapay zekâ ve derin öğrenme gibi yoğun veri işleme uygulamalarında artmaktadır (Greenberg-Toledo et al., 2019). Ancak, mevcut platformların DNN'ler için verimli olmadığı bilinmektedir. Bu sorunların üstesinden gelmek amacıyla, DNN tabanlı uygulamalara yönelik özel donanım tasarım çalışmaları yürütülmektedir. Bu durumdan yola çıkan Greenberg-Toledo ve arkadaşları, çalışmalarında yapay sinir ağlarının temel unsuru

olan sinapsı oluşturmak için memristörleri kullanmışlardır. Çalışmada, DNN'lerde yaygın olarak kullanılan momentum optimizasyon algoritmasını destekleyen bir memristör tabanlı sinaps önerilmiştir. Simülasyonlarda, önerilen DNN eğitim çözümlerinin performansı ortalama 886 kat hızlandırdığı ve enerji tüketimini ortalama yedi kat azalttığı gözlemlenmiştir. Ayrıca, bir GPU platformunda eğitim kadar doğru olduğu da görülmüştür.

Derin sinir ağı modellerinin memristör tabanlı çapraz çubuklar gibi nöromorfik hızlandırıcılarla kullanılması durumunda, kullanılan donanım birimlerinin güvenilirliğini sağlamak öncelikli konulardan biridir (Du et al., 2015). Üretim sürecindeki hatalar ve değişkenlikler, çapraz çubuk bağlantılarında donanım arızalarına yol açabilir (C. Y. Chen & Chakrabarty, 2021). Chen ve Chakrabarty, çalışmalarında, memristör çapraz çubuklarında yanlış sınıflamaya neden olabilecek Kritik Hataları (CF'ler) belirlemek için makine öğrenimi tabanlı verimli bir teknik geliştirmişlerdir. Bu önemlidir çünkü memristör tabanlı DNN'ler doğal olarak hata toleranslıdır ve CF'leri doğru bir şekilde tespit etmek, hata toleransı alternatiflerinin bu alanlara odaklanmasına olanak tanır (C. Y. Chen & Chakrabarty, 2021). Bu teknik, rastgele hata enjeksiyonu kullanarak %98'in üzerinde doğrulukla ve 20 kat daha hızlı bir şekilde kritik hataları tespit edebilmektedir. Bu çalışmada, memristör çapraz çubuklarında CF'lerle ilişkili ağırlıkları kaldıran bir hata toleransı yöntemi önerilmektedir. CIFAR-10 veri kümesi ve diğer derin sinir ağları ile yapılan testlerde, önerilen budama tekniğinin bağlantı alanlarının %95'ine kadarını kaldırdığı ve DNN çıkarım doğruluğunda %1'den az bir düşüşle sonuçlandığı gösterilmiştir. Bu, genel CF sayısındaki azalma sayesinde hata toleransı için gereken fiziksel donanım yedekliliğinde %99'luk bir azalma sağlamıştır.

Yapay zekâ, bulut bilişim ve Nesnelerin İnterneti (IoT) uygulamalarının hızla büyümesiyle, derin öğrenme uygulamalarında hesaplama için memristör cihazları ve ilgili donanım sistemlerinin geliştirilmesi, düşük güç tüketimi ve daha az çip alanıyla kapsamlı veri hesaplamaları gerektirmektedir. Derin öğrenme modelleri, düşük güç tüketimi ile çalışabilmek için büyük miktarda veri işlemeyi gerektirir. Yapılan bir çalışma (KI & R, 2022) ile CIFAR-10 veri kümesi için memristör tabanlı bir nesne algılama sistemi önerilmiş ve bu sistemle %85 doğruluk oranına ulaşmayı başarmışlardır.

Memristörler, bellek birimleri üzerinde matris-vektör çarpımlarını (MVM) hesaplamayı mümkün kılar (Kern et al., 2022). Memristörler, derin sinir ağı hızlandırıcılarının enerji verimliliğini önemli ölçüde artırma potansiyeline sahiptir. Ancak, memristörlerdeki hesaplamalar donanım uyumsuzluk sorunlarına ve çeşitli gürültü kaynaklarına maruz kalmakta olup, bu durum sistem performansını olumsuz etkileyebilir. Kern ve arkadaşları, yaptıkları

çalışma (Kern et al., 2022) ile matris-vektör çarpımı (MVM) için memristör çapraz çubuklarının kullanıldığı durumda derin sinir ağlarının (DNN) ortalama kare hata (MSE) değerine yönelik teorik bir analiz gerçekleştirmişlerdir. Bu çalışmada, hem DNN modelinin boyutunun küçültülmesi ihtiyacından kaynaklanan kuantizasyon gürültüsünü hem de bellek değerini programlarken ortaya çıkan değişkenlik nedeniyle oluşan programlama gürültüsünü dikkate alan yeni bir yöntem önerilmiştir. Önerilen yöntemin, Monte-Carlo simülasyonuna kıyasla yaklaşık iki kat daha hızlı olduğu gözlemlenmiştir (Kern et al., 2022). Bu araştırmanın sonucu, belirli bir güç sınırı ve minimum hata ile uyumlu olacak şekilde uygulama parametrelerini optimize etmeyi mümkün kılmaktadır.

Memristör tabanlı yapay nöron ve sinaps

Memristör tabanlı nöromorfik hesaplama sistemleri, Sinir Ağı (NN) algoritmalarına enerji açısından daha verimli bir alternatif sunmaktadır (Ren et al., 2022). Doğal analog dirençler olan memristörler, hesaplamayı bellekte gerçekleştirerek von Neumann mimarisine bir alternatif sunmaktadır (Jo et al., 2010). Memristörlerin büyük ölçekli uygulamalarında optimum performansı sağlamak için, tasarımı tahmin etmek ve optimize etmek amacıyla sistemin simülasyonları yapılmalıdır.

Nöromorfik hesaplama, biyolojik olarak ilham alarak ve insan beyninin çalışma şeklini modelleyerek zorlu yapay zekâ ve makine öğrenimi problemlerini çözmek için kullanılabilecek yüksek seviyeli bağlantı modellerine sahip nöronlar ve sinapslar oluşturmayı amaçlamaktadır (Ren et al., 2022). Yapay Sinir Ağı algoritmaları, insan beyninin nöron ağı yapısının çalışma şeklini taklit ederek hızlı hesaplamalar yapabilmektedir (Mead, 1990). Paralel hesaplamada, sinir ağlarındaki işlemleri hızlandırmak için bir ağırlık matrisi kullanılır. Yapay zekâ uygulamaları için tasarlanan çiplerin büyük çoğunluğu hızlandırıcı olsa da bunlar nöromorfik işlemciler değildir (Du et al., 2015). Yapay zekâ ve derin öğrenme alanındaki büyük şirketlerden bazıları Google, Microsoft, IBM, Amazon, Apple, Intel, NVIDIA ve Qualcomm'dur. Yapay zekâ alanında çalışan bu büyük şirketler, Grafik İşleme Birimi'nden (GPU) yararlanarak matris çarpımı gibi işlemler için hızlandırıcılar, örüntü tanıma gibi önemli uygulamalar için Alan Programlanabilir Kapı Dizileri (FPGA) veya belirli görevler için Uygulamaya Özel Entegre Devreler (ASIC) gibi dijital entegre devreler geliştirmeye çalışmaktadır (Capra et al., 2020). Bu çalışmalarla birlikte, yüksek çip fiyatları, donanım rekabeti, hesaplama hızı, düşük güç tüketimi, küçük ayak izi boyutu ve düşük üretim maliyetine odaklanmaktadırlar (Schuman et al., 2017). Hızlandırıcıların önemli ve temel işlevlerinden biri matris çarpımıdır (Sung et al., 2018). Google, makine öğrenimi görevlerini hızlandırmak için

Tensör İşleme Birimi (TPU) adı verilen kendi yapay zekâ çipini geliştirmektedir (Jouppi et al., 2017). Çok hızlı veri işleme ve hesaplamaya sahip olan TPU'da ana hesaplama kısmı aynı zamanda bir matris çarpma birimidir (Y. E. Wang et al., 2019). Nöromorfik hesaplama, insan beyninin çalışma şeklini taklit ettiği için çok katmanlı nöronal ağların her bir parçasının modellenmesi ve derin öğrenme algoritmalarının donanım birimine uygulanması ile gerçekleştirilmektedir (Schuman et al., 2017). Nöromorfik hesaplama modelini temel alan CMOS teknolojileri ile üretilen yapay sinir işlemcileri bulunmaktadır. Bu işlemciler biyolojik nöronların davranışlarını taklit edecek şekilde tasarlanmıştır ve yapay sinir ağları oluşturmak için kullanılabilmektedir. Ancak geliştirilen CMOS tabanlı sinir ağının verimli olabilmesi için memristörlerin bir çip üzerinde denetimli eğitim ve öğrenme için kullanılabilmesi gerekmektedir (Sung et al., 2018).

Memristif cihazlar, çoklu direnç seviyelerini depolama yetenekleri nedeniyle matris çarpma birimleri için çok uygundur (Y. E. Wang et al., 2019). Memristörler, verileri depolamak ve işlemek için kullanılabilen bir tür uçucu olmayan bellek cihazıdır. Memristörler, düşük güç tüketimi ve yüksek yoğunluğa sahip olmaları nedeniyle nöromorfik donanım için caziptir (Hung et al., 2021). Örüntüleri tanımak, görüntüleri işlemek ve diğer makine öğrenimi görevlerini yerine getirmek için yapay sinir ağları oluşturmak için kullanılabilirler. Ayrıca, sinyal işleme ve kontrol uygulamaları için analog devreler oluşturmak için de kullanılabilirler. Memristörler, çip üzerinde hesaplama için nöromorfik donanım oluşturmak için de kullanılabilir ve verilerin daha hızlı ve daha verimli işlenmesine olanak tanır. Örneğin memristör, eşik anahtarlama öğrenimi (Yakopcic et al., 2011) kullanılarak oluşturulan tek bir başak ve salınım hareketine sahiptir (Pickett et al., 2013). Memristif sinaptik ağırlık, hafıza değiştirme özelliğinden kaynaklanmaktadır (Woo et al., 2016; J. J. Zhang et al., 2013; W. Zhang et al., 2023). Ayrıca memristif mantık (Maan et al., 2017) ve memristör tabanlı tanıma çipi de bulunmaktadır.

Memristör tabanlı donanım üzerine yapılan çalışmalarda, özellikle sinaptik ağırlıklarda, veri saçılması ve güvenilirlik problemlerini çözmek için yoğun bir çaba vardır (Sung et al., 2018). Bununla birlikte, bu sorunları çözmek için çeşitli yöntemler geliştirilmektedir. Bunlar arasında memristörleri kontrol etmek için kullanılan kontrol sinyallerinin düzenlenmesi, memristörlerin çalışma ortamının kontrol edilmesi, memristörlerin kalibre edilmesi ve memristörler üzerinde çalışan algoritmaların geliştirilmesi yer almaktadır (Rajendran et al., 2019). Bu çalışmaların sonuçları, memristör tabanlı donanımın ticari bir ürün haline gelmesi için gereken güvenilirlik ve veri saçılımı önündeki engellerin kaldırılmasına yardımcı olacaktır.

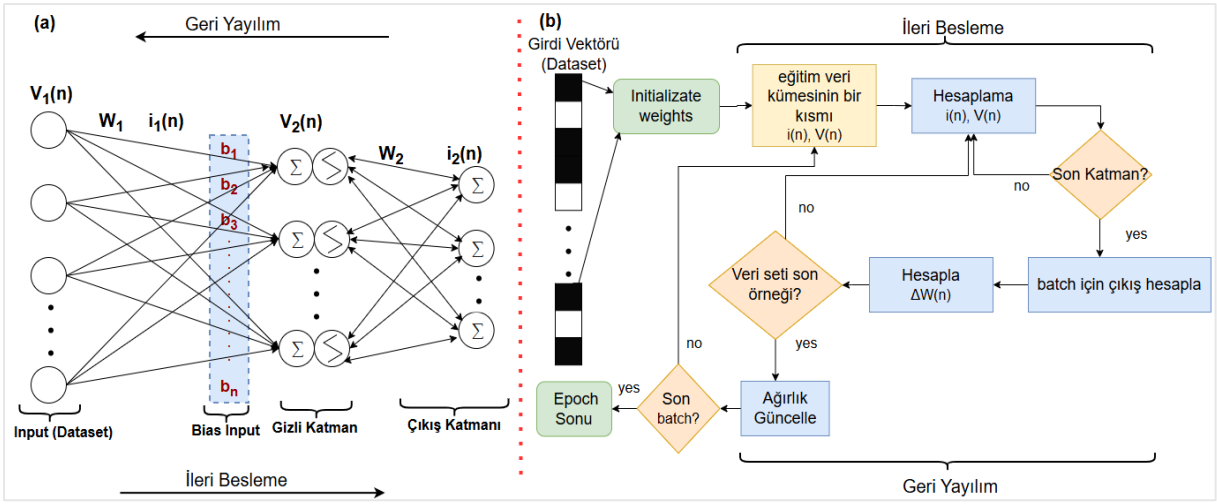
Bu, verilerin kendisinin rastgelelik veya diğer faktörlere atfedilebilecek bir dereceye kadar değişkenliğe sahip olacağı anlamına gelir. Bu değişkenlik, zaman içinde farklı modeller veya eğilimler gösterebileceğinden anahtarlama verilerinde görülebilir (Sung et al., 2018). Örneğin, olasılık teorisi kapsamında Poisson dağılımı ((Langston et al., 2011), belirli bir ortalama değer etrafında rastgele meydana gelen olayların sayısının dağılımını temsil eder. Bu dağılım, rastgeleliğin kontrol edilemediği anahtarlama süreci sırasında iletken yolların oluşumu için de geçerlidir (Yoo et al., 2008). Bu engellerin üstesinden gelmek için araştırmacılar çeşitli çözümler önermişlerdir. Yaklaşımlardan biri memristörler ile dirençler, kapasitörler ve transistörler gibi diğer bileşenleri bir arada kullanarak hibrit bir devre oluşturmaktır (Borghetti et al., 2009). Bu hibrit devre, verileri tek başına bir memristörden daha güvenilir bir şekilde depolamak ve işlemek için kullanılabilir. Ayrıca, araştırmacılar memristör tabanlı sistemlerin güvenilirliğini artırmak için makine öğrenimi algoritmalarının kullanılmasını önermişlerdir (Ren et al., 2022). Bu algoritmalar, memristörde depolanan verilerdeki hataları tanımlamak ve düzeltmek için kullanılabilir ve daha güvenilir bir çalışma sağlar. Son olarak, araştırmacılar memristör tabanlı sistemlerin güvenilirliğini artırmak için hata düzeltme kodlarının kullanılmasını önermiştir (Ren et al., 2022; Schuman et al., 2017). Bu kodlar, memristörde depolanan verilerdeki hataları tespit etmek ve düzeltmek için kullanılabilir ve daha güvenilir bir çalışma sağlar (Z. Wang et al., 2018). Yapılan çalışmalar sonucunda güvenilir bir kalıcı doğrusal çok seviyeli memristör henüz ortaya çıkarılamamıştır. Bununla birlikte, bir memristör koleksiyonu, çok seviyeli verilerin varyansını azaltmaya yardımcı olabilecek çoklu bit veya çoklu seviyeli bir sinaptik ağırlık uygulamak için hala çalışmaktadır.

Memristör nöronlar

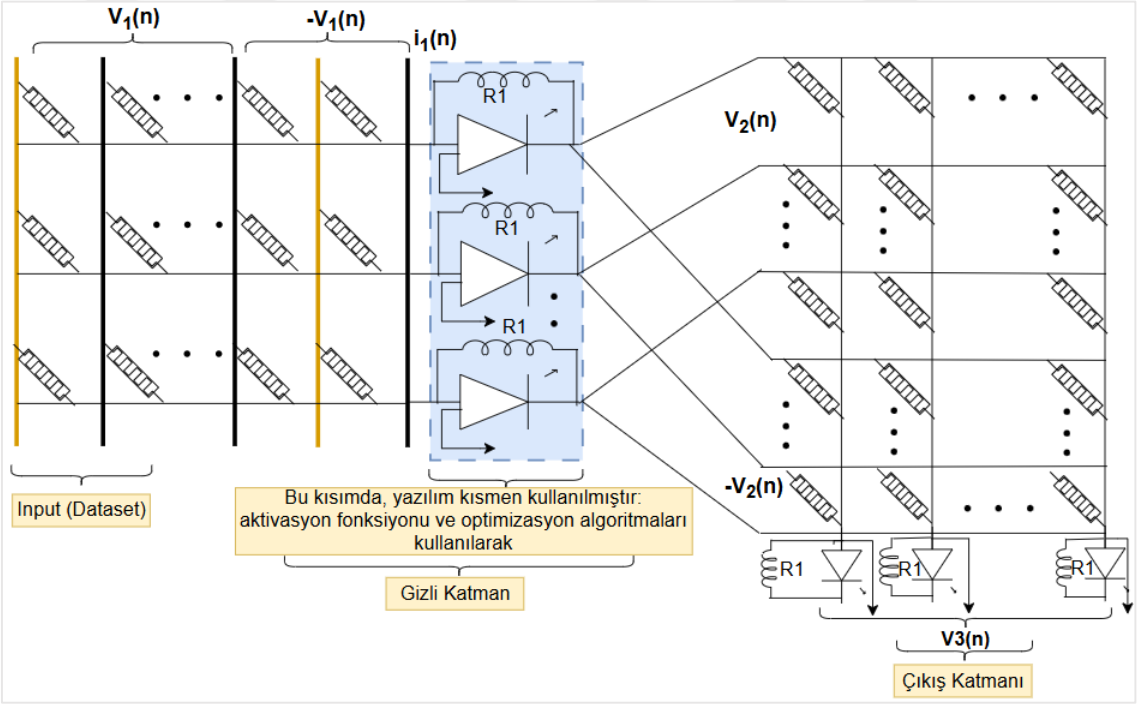
Günümüzde yapay nöromorfik sistemler, insan beynini daha iyi anlamamızda ve nörolojik süreçleri taklit eden bilgi işlem sistemlerinin geliştirilmesinde önemli bir rol oynamaktadır. Biyolojik sistemlerde gözlemlenen uzamsal ve güç verimliliğine ulaşmak için, sinapsların nano ölçekli analogları olarak elektronik ve faz değişimli memristif cihazlar keşfedilmiştir (Tuma et al., 2016). Araştırmalar, biyolojik nöronların davranışını taklit eden yapay nöronların, kalkojenit bazlı faz değişim malzemeleri kullanılarak inşa edilebileceğini göstermiştir. Bu yapay nöronlar, nano ölçekli faz değiştirme cihazlarının faz konfigürasyonu ile temsil edilmektedir (Tuma et al., 2016).

Bir sinir ağı modelinin genel yapısı, ağın hangi bileşenlerden oluştuğunu, bu bileşenlerin çalışma prensibini ve bu bileşenler arasındaki etkileşimi açıklar (Y. Shi et al., 2019) Örneğin, Şekil 9'da gösterilen yapay sinir ağı modelinin ortak bileşenleri, Şekil 12'de gösterilen biyolojik

sinir ağı modelinin yapısından esinlenerek nöronlar ve sinapslardır. Ayrıca, bir sinir ağı modeli tasarlanırken, ağı oluşturan her bir bileşen için modeller tanımlanmalı ve tanımlanan modele göre bileşenin nasıl çalışacağı belirlenmeli ve yönetilmelidir (Haoxiang & S, 2021). Memristör modeli, gerçek bir memristörün davranışını matematiksel olarak temsil etmek için geliştirilmiştir ve özelliklerini araştırmak için kullanılabilir (X. Liu & Zeng, 2022). Sistem tasarımını hızlandırmak için simülasyonlarda da kullanılabilir. Nöronlar sadece CMOS teknolojisi ile üretilirken, çok sayıda transistöre ihtiyaç duyulmaktadır. Memristörler, devreleri basitleştirmek için bazı CMOS cihazlarına alternatif olarak önerilmektedir. Önerilen memristör tabanlı uygulamalara örnek olarak; memristör tabanlı Hodgkin-Huxley (Corinto et al., 2013), memristör tabanlı Morris ve Lecar (Shamsi et al., 2017), memristör tabanlı Fitz Hugh-Naguma (J. Zhang & Liao, 2017) ve memristör tabanlı Hindmarsh-Rose (Bao et al., 2018) sinyalleri simüle edilerek rapor edilmiştir. Basit memristör tabanlı başak örüntüsü, nöronların tutumunu simüle etmek için memristörleri kullanan matematiksel bir modeldir (Zheng & Mazumder, 2018). Memristörlerin, elektrik sinyalleri üretme ve yayma yetenekleri gibi nöronların elektriksel özelliklerini temsil etmek için kullanılabileceği fikrine dayanmaktadır. Model, sinir ağlarının dinamiklerini incelemek ve makine öğrenimi için yeni algoritmalar geliştirmek için kullanılır. Entegre et ve ateş modeli, nöronların dinamiklerini birleştiren biyolojik olarak esinlenmiş bir modeldir (Lashkare et al., 2018). AlShedivat ve meslektaşları, bir memristör kullanarak stokastik olarak artan bir nöronu simüle etmiştir (Al-Shedivat et al., 2015). Bilim insanları memristörlerin gelişmiş analitik modelini önermişlerdir. Memristörler için önerilen analitik model, memristörün direncini tanımlayan bir "durum değişkeni" fikrine dayanıyordu (Strukov et al., 2008). Memristörün direnci doğrusal değildir ve kendisine uygulanan akım ve voltaj tarafından belirlenir, bu da bu durum değişkeninin modeline yansıtılır. Model daha sonra gerilim darbesi, akım darbesi ve sinüzoidal dalga formu gibi çeşitli koşullar altında bir memristörün davranışını simüle etmek için kullanılmıştır. Simülasyonların sonuçları, modelin bu koşullar altında memristörün davranışını doğru bir şekilde tahmin ettiğini göstermiştir. Shamsi ve arkadaşları, memristör tabanlı analog uyarlanabilir bir nöron tasarlamıştır (Binelli et al., 2005). Mehonic ve Kenyon, tek kutuplu anahtarlama belleği SiO_2 'ye bir eşik akımı uygulayarak sınır voltajının yükselişini/kararsızlığını izlemiştir (Mehonic & Kenyon, 2016). Pantazi ve arkadaşları, faz değişimli memristörleri içeren bir mimari geliştirerek nöronların entegrasyonu ve aktivasyonunun yanı sıra sinaptik unsurların zaman içinde gelişmesine olanak sağlamıştır (Pantazi et al., 2016).



Şekil 12. (a) MLP'nin matematiksel modeli. **(b)** MLP'yi ayrıntılı olarak açıklayan akış diyagramının gösterimi.



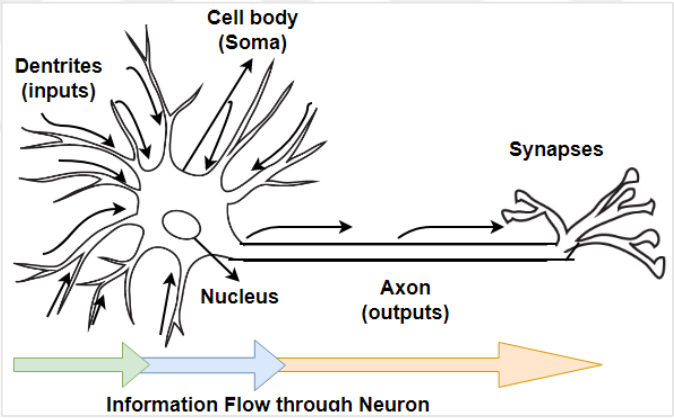
Şekil 13. MLP'nin memristör tabanlı uygulamasının şematik gösterimi.

Bilgi işleme donanımı, yapay zekânın günlük yaşam üzerindeki etkilerini belirleyen temel bir unsurdur. Ancak, derin sinir ağlarını çalıştıran mevcut donanımların enerji tüketimi, biyolojik beyinle kıyaslandığında oldukça yüksektir (Huang et al., 2023; Jeong et al., 2016). Tek katmanlı algılayıcılar sınırlı işlevselliğe sahip olup, bu kısıtın üstesinden gelmek amacıyla çok katmanlı algılayıcılar (MLP'ler) geliştirilmiştir. Şekil 12(a)'da gösterildiği üzere, MLP'ler giriş vektörlerini çıkış vektörlerine eşleyen ileri beslemeli sinir ağlarıdır ve daha yüksek doğruluk gerektiren karmaşık görevlerde etkili bir şekilde kullanılmaktadır (Ruck et al., 1990; Thimm & Fiesler, 1997). Şekil 12(b), Şekil 12(a)'da matematiksel olarak tanımlanan MLP'nin

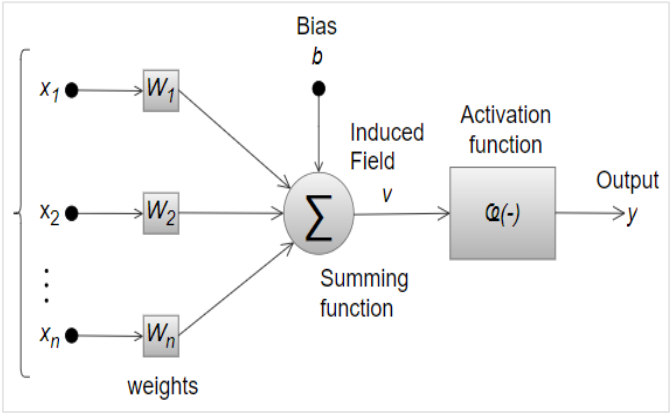
akış diyagramını göstermektedir. Şekil 13'te ise, Şekil 12(a)'da gösterilen MLP'nin elektrik devre bileşenleriyle fiziksel bir analojisi sunulmaktadır. Bu analojide, giriş katmanı veri kümesinden alınan girdileri işlerken, bu sinyaller gizli katmanlar aracılığıyla dirençler ve yükselteçler gibi devre elemanlarından geçmektedir. İşlenmiş sinyallerin çıkış katmanına ulaşmasıyla nihai sonuçlar elde edilmektedir. Bu devre elemanları sinyal işleme ve dönüştürme süreçlerini simüle etmektedir. Ancak, bu algoritmalar paralel işlem ve sürekli bilgi akışı için uygun olmayan von Neumann mimarisi nedeniyle verimsiz çalışmaktadır (Han et al., 2022; S. H. Sung et al., 2021; Zanotti et al., 2021). Bu bağlamda, donanım tabanlı sinir ağları ya da nöromorfik işlemciler, biyolojik nöronların işleyişini taklit ederek performans artışı sağlamaktadır (Y. Liang et al., 2022; Xiao et al., 2020) Nöromorfik sistemler, sinyal iletim katsayısının "ağırlık" olarak işlev gördüğü yapay sinapslar üzerinden işlem yapmaktadır (Onen et al., 2020). Ayrıca Şekil 13'te gösterilen şema ile çalışma kapsamında önerilen memristör tabanlı modelin donanım kısmında uygulanması gösterilmiştir. Burada yapay sinir ağı yapısında bulunan her bir nöronun içerisine bir memristif yapı eklenerek modelin tasarımı gerçeklenmiş ve bilgisayar ortamında simüle edilmiştir. Bu şema, memristör tabanlı bir yapay sinir ağını (YSA) temsil eder. Giriş katmanında $V_1(n)$ ve $-V_1(n)$ sinyalleri memristör ağı üzerinden geçirilir. Memristörler, sinir ağı ağırlıklarını temsil eder ve dirençleri önceki durumlarına bağlı olarak giriş sinyallerini işler. Gizli katmanda, aktivasyon fonksiyonları ve amplifikatörler yardımıyla sinyaller doğrusal olmayan şekilde düzenlenir ve işlenir. Çıkış katmanında ise işlenmiş sinyaller birleştirilerek nihai çıktı oluşturulur. Memristörler sayesinde hem bellek hem de işlem bir arada gerçekleştirilir. Bu, düşük enerji tüketimi ve hızlı hesaplama avantajı sağlar. Çalışma kapsamında önerilen modelin memristör tabanlı donanım-yazılım geçişi "MATERYAL ve METOT" kısmında "Yapay sinaps olarak memristör ve uygulamanın yazılım-donanım entegrasyonu" alt başlığında detaylı biçimde anlatılacaktır.

Şekil 14'de gösterildiği gibi, biyolojik bir nöron genellikle bir hücre gövdesi, bir akson ve dendritlerden oluşur (G. Lee et al., 2021). Akson, bir nöronun hücre gövdesinden uzanan uzun, ince bir liftir ve elektrik sinyallerini nörondan diğer hücrelere taşımaktan sorumludur (Lv et al., 2018). Dendritler, sinyalleri almaktan, entegre etmekten ve hücre gövdesine iletmekten ve diğer sinir hücrelerinden girdi almaktan sorumlu olan dallardır. Kimyasal iletimler, nörotransmitterler bir nörondan salındığında ve başka bir nöron üzerindeki reseptörlere bağlandığında meydana gelir ve bir elektrik sinyalinin gönderilmesine neden olur (Gul, 2020). Elektriksel iletimler, nöronlar iyonların doğrudan aralarında akmasına izin veren ve bir elektrik akımı oluşturan boşluk kavşakları ile bağlandığında meydana gelir (Schuman et al., 2017).

Sinapslar, elektriksel veya kimyasal sinyallerin bir nörondan diğerine iletildiği iki nöron arasındaki bağlantı noktalarıdır (H. Wang et al., 2018; S. Yu et al., 2011). Elektrik sinyalleri gönderen nöronun aksonu tarafından üretilir ve kimyasal sinyaller gönderen nöronun akson terminalinden salınır ve alıcı nöronun dendritleri tarafından alınır (G. Lee et al., 2021). Nöron daha sonra sinaps boyunca diğer nöronlara giden nörotransmitterleri serbest bırakır ve böylece sinyali yayar (Dretchen et al., 1976). Bu süreç aksiyon potansiyeli olarak bilinir. Nöron yeterince sinyal alırsa, bir eşiğe ulaşacak ve bu noktada bir aksiyon potansiyeli ateşleyerek diğer nöronlara nörotransmitter salacaktır (X. Zhang et al., 2018). Bazı nöromorfik modellerde, yük birikimi, yükü depolayan ve nöron ateşlenmeden önce belirli bir miktarda yük birikmesine izin veren kapasitörlerin kullanılmasıyla elde edilir. Bu tür bir yük birikimi, spiking nöronları simüle etmek için popüler bir model olan Leaky Integrate-and-Fire (LIF) modelinde kullanılmaktadır (Lu et al., 2020). Bu modelde, kapasitör bir akım girişi ile şarj edilir ve daha sonra zaman içinde yavaşça boşalır. Kapasitör belirli bir eşiğe ulaştığında, nöron ateşlenerek diğer nöronlara bir sinyal gönderir (Shamsi et al., 2018). Biyolojik bir nöronun yapısı ve elektrik sinyallerinin iletimi Şekil 14'de gösterilmektedir.



Şekil 14. Biyolojik Nöron üzerinden bilgi akışı



Şekil 15. Yapay Nöron

Memristör sinapsları

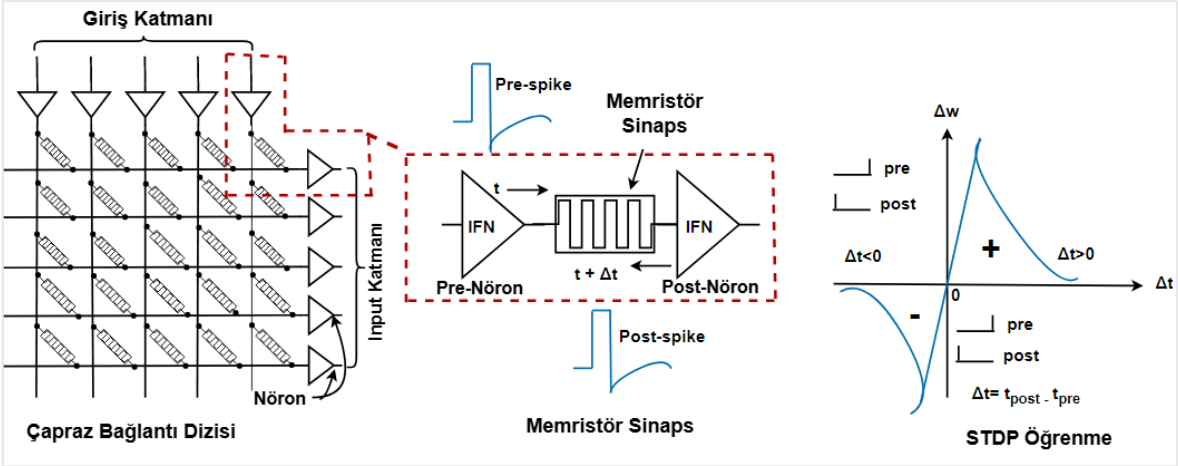
Nöromorfik çalışmaların bazılarında nöron modeline odaklanılarak sinaps uygulamalarını da içeren bir sistem geliştirilebilmektedir. Bu duruma benzer şekilde nöron modellerinden bağımsız sinaps uygulamaları geliştirmeye odaklanan çalışmalarda da nöromorfik sistemler gerçekleştirilmiştir (Merolla et al., 2011; J. Seo et al., 2011). Bu odaklanmalarla birlikte memristör sinaptik ağırlıklarında karmaşık ve zorlayıcı durumlar söz konusudur. Bu zor durumlar; kalıcılık, doğrusallık ve çok düzeyliliktir (Amirsoleimani et al., 2020; Huh et al., 2020). Ancak bu üç özelliği aynı anda sağlayan sonuçlar henüz elde edilememiştir (Huh et al., 2020). Sinaps, biyolojik sinir ağlarının önemli bir parçası olmakla birlikte doğrudan elektronik bir eşdeğeri yoktur (Thomas, 2013). Bu durum, sinir sisteminin yapısını taklit eden donanımların oluşturulmasını zorlaştırmaktadır. Şekil 13, bir nöron darbesi üreten bir CMOS entegre ve ateşleme nöronunu göstermektedir (Xinyu Wu et al., 2015). Memristör tabanlı bir sinaptik ağırlık çapraz çubuğunda, memristör giriş ve çıkış nöronları arasına yerleştirilir ve aralarındaki bağlantının ağırlıklarının ayarlanmasına ve saklanmasına izin verir.

Birçok çalışma, spike zamanlamasına bağlı plastisite (STDP) gibi temel sinaptik öğrenme ilkelerinin memristörlerde başarıyla uygulandığını göstermiştir (Alibart et al., 2012; Jo et al., 2010; Krzysteczko et al., 2012; S. Yu et al., 2011). Daha önceki araştırmalarda, memristörler öncelikle programlanabilir bellek bileşenleri olarak görülmüş ve kullanılmıştır. Oksit tabanlı memristörlerde ikinci dereceden memristör etkilerinin önemli olabileceği deneysel olarak kanıtlanana kadar, çalışmalar genellikle karmaşık programlama dalga formları veya üst üste binen programlama darbeleri kullanan tasarımlara odaklanmadı (S. Kim et al., 2015). Bu cihazlarda, istenen iletkenlik değişimini elde etmek için darbe genişliği ve darbe yüksekliğinin dikkatli bir şekilde tasarlanması gerekir. Matematiksel açıdan bu aygıtlar birinci dereceden memristörler olarak sınıflandırılabilir (Pershin & Di Ventra, 2012).

Çapraz çubuk üzerindeki ikinci dereceden memristörlerde, her memristör STDP adlı bir öğrenme kuralı kullanılarak bir eğitim sürecine tabi tutulur (Cruz-Albrecht et al., 2012). STDP öğrenimi sinaptik ağırlığı, nöronun sinapstan önce ateşlediği zaman ile sinapstan sonra ateşlediği zaman arasındaki zaman farkına göre belirler. Bu durumda, sinaptik ağırlık doğrusal olmayan sonuçlar üretir ve genellikle Winner-Take-All (WTA) algoritması ile denetimsiz öğrenmede kullanılır (Sung et al., 2018). Zheng ve Mazumder tarafından önerilen algoritma, spike-zamanlamaya bağlı plastisite (STDP) ve denetimli öğrenmenin bir kombinasyonunu

kullanan bir tür makine öğrenimi veya derin öğrenme olan denetimli STDP öğrenme kavramına dayanmaktadır.

Bilim insanları, termal enerjinin dağılımından yararlanarak kimyasal sinapslarda gözlemlenen kalsiyum iyonu (Ca_2^+) dinamiklerini taklit etmek için çalışmalarında ikinci dereceden memristörleri entegre ettiler. Bu yaklaşım sayesinde, üst üste binmeyen spike uçlarla spike zamanlamasına bağlı plastisite (STDP) olarak bilinen olguyu etkili bir şekilde sergilediler ve diğer çeşitli sinaptik işlevleri çoğaltmayı başardılar. Bu başarı, biyolojik olarak doğru sinaptik cihazların geliştirilmesine yönelik kayda değer bir ilerlemeye işaret etmektedir. Bu yaklaşım tekrarlanabilirlik ve basitlik özelliklerine sahiptir ancak fiziksel süreçlerin gerçek sinapslardan önemli ölçüde farklı olması nedeniyle istenen sinaptik işlevlerin doğruluğu ve çeşitliliği sınırlıdır (Z. Wang et al., 2017). Biyolojik Ca_2+ dinamiklerinin fiziksel özelliklerini kopyalayan bir cihaz oluşturmak, sinaptik işlevi daha etkili bir şekilde taklit etme yeteneğini artıracak ve nöromorfik bilgi işlemin potansiyel uygulamalarını genişletecektir. Bu emülatif memristör, metal atom difüzyonu ve nanoparçacıkların kendiliğinden oluşumu mekanizmalarını kullanarak çalışmaktadır. Bu mekanizma, yüksek çözünürlüklü transmisyon elektron mikroskobu (HRTEM) ve nanoparçacık davranışının dinamik simülasyonları ile derinlemesine incelenmiştir. Difüzyon memristörlerinin dinamik özelliklerinin, kısa ve uzun vadeli plastisite gibi operasyonel özellikler de dahil olmak üzere biyolojik sinapslardaki Ca_2+ ile işlevsel olarak eşdeğer olduğu deneysel olarak doğrulanmıştır (Z. Wang et al., 2017). Memristör sinapslarına sahip Crossbar SNN mimarisi Şekil 16'te gösterilmektedir.



Şekil 16. Memristör sinapslı Crossbar SNN mimarisi, insan beyninin çalışma şeklinden esinlenen çift yönlü bir STDP öğrenme kuralının grafiksel bir temsilinin eşlik ettiği iki sinaptik öncesi spike ve iki sinaptik sonrası spike arasındaki bir sinaps bağlantısıdır. Verileri depolamak ve işlemek için çapraz çubuk memristör dizileri kullanan bir tür nöromorfik hesaplama mimarisidir.

Algoritma donanım uyumlu olacak şekilde tasarlanmıştır, yani önemli değişiklikler gerektirmeden mevcut donanım platformlarında uygulanabilir. Algoritma ayrıca ağırlığa bağlı olacak şekilde tasarlanmıştır, yani öğrenme süreci nöronlar arasındaki bağlantıların ağırlıklarına dayanmaktadır (Zheng & Mazumder, 2018).

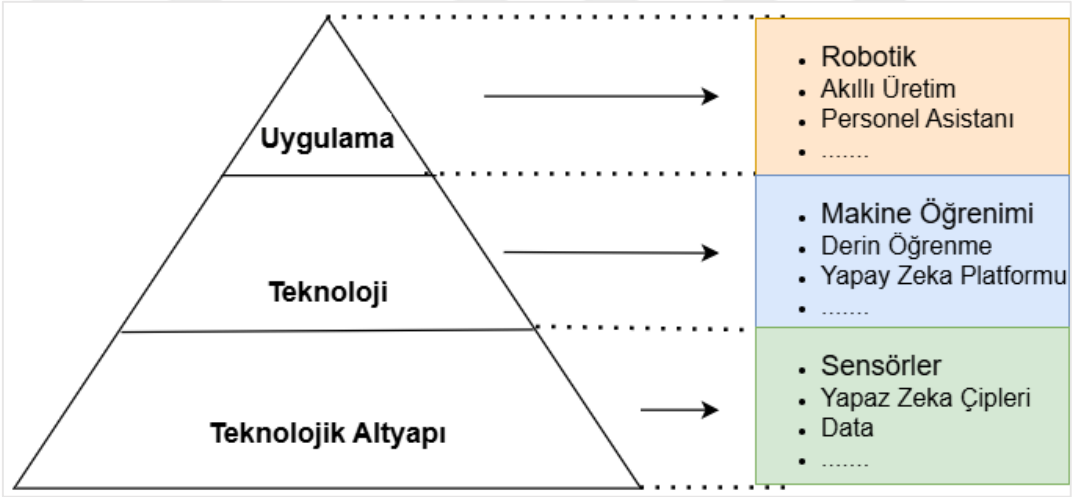
Araştırmacılar analog bellek özellikleri oluşturmak amacıyla nöronlar, sinapslar, mimari, öğrenme ve test üzerine bir dizi çalışma yürütmüş ve yaygınlaştırmıştır (Tsai et al., 2018). Genel olarak sinaps modelleri iki kategoride incelenebilir. İlk kategori, biyolojik beynin yapısından esinlenen ve spike tabanlı sistemler için sinaps yapılarını içeren sinaps uygulamalarıdır (Q. Wan et al., 2019). İkinci kategori ise ileri beslemeli sinir ağları gibi geleneksel yapay sinir ağları için sinaps uygulamalarıdır (Zyarah et al., 2017). Nöromorfik sistemlerde en bol bulunan bileşenin sinapslar olduğu belirtilmektedir. Geliştirilecek çoğu donanım uygulaması için sinaps uygulamasının optimize edilmesine, özellikle nöromorfik hesaplama sistemleri için yeni malzemeler geliştirilmesine odaklanılmaktadır (Kwon et al., 2022). Biyolojik sistemlerin ayrıntılı ve hassas bir şekilde donanım modellemesi çok zordur. Bu nedenle, biyolojik davranışı açıkça modellemeye çalışılmadığı sürece sinaps modelleri nispeten basit olma eğilimindedir (Schuman et al., 2017). Nöronun ağırlık değerinin değişmesine neden olan bir plastisite mekanizması vardır. Ayrıca, plastisite mekanizmalarının biyolojik beyinlerde öğrenme ile ilişkili olduğu bulunmuştur.

Memristör tabanlı yapay zekâ (AI) çipleri

AI çipleri, AI yeteneklerini içeren ve makine öğrenimi alanında kullanılan sofistike silikon mikroişlemcilerdir. Yapay zekâ, çeşitli sektörlerde insan hayatına yönelik potansiyel tehlikelerin azaltılmasında veya hafifletilmesinde çok önemli bir rol oynamaktadır. Artan veri hacmi, matematiksel ve hesaplama zorluklarının üstesinden gelmek için daha verimli sistemlerin geliştirilmesini gerektirmekte ve daha fazla üretkenlik için artan aciliyeti vurgulamaktadır. Bu nedenle, yapay zekâ çiplerinin ve uygulamalarının geliştirilmesi söz konusu olduğunda, BT sektöründeki büyük firmaların önemli bir kısmı kendilerini bu işe adamış durumdadır.

Yapay zekâ özellikli uç cihazların yüksek çıkarım doğruluğuna, hızlı tepki sürelerine ve enerji tasarruflu çalışmaya sahip olması, yani uzun ömürlü pillere sahip olması gerekir (Chiu et al., 2023). Bu tür cihazların halka açık alanlarda kullanılması, onları çip kontrolünde veya çip üstü kalıcı bellekte depolanan değerli verilere yetkisiz erişimi içeren kötü niyetli saldırılara karşı savunmasız hale getirmektedir (Golonzka et al., 2019; T.-H. Yang et al., 2018).

Milyonlarca parametreye sahip yapay zekâ (AI) modelleri çeşitli görevlerde yüksek hassasiyet elde edebilir (Vaswani et al., 2017), ancak grafik işlem birimleri veya merkezi işlem birimleri gibi geleneksel genel amaçlı işlemciler gibi donanım bileşenlerinin zayıf enerji verimliliği daha fazla sorun haline gelebilir (Ambrogio et al., 2023). Genellikle analog-AI olarak adlandırılan analog bellek içi hesaplama (Khaddam-Aljameh et al., 2022; Narayanan et al., 2021; W. Wan et al., 2022; Yao et al., 2020), matris-vektör çarpımlarını "bellek dizileri" içinde eşzamanlı olarak yürütme yeteneği sayesinde üstün enerji verimliliği elde etmektedir (Ambrogio et al., 2023). Ambrogio ve meslektaşları çalışmalarında, 34 farklı birimdeki 35 milyon faz değişimli bellek cihazını, birimler arası kitlesel paralel iletişimi ve analog, düşük güçlü çevresel devreleri bir araya getiren analog bir yapay zekâ çipi sunmuştur. Yapay zekâ çipinin çeşitli yapay zekâ katmanları içindeki işlevselliği Şekil 17'te gösterilmektedir.



Şekil 17. Yapay zekanın çeşitli katmanları içinde bir yapay zekâ çipinin işlevi.

Memristör tabanlı yapay sinaps çalışmaları

Bilgisayar sistemlerinin gelişim süreci incelendiğinde, hesaplama işlemleri için veri miktarı katlanarak arttığından bellek birimleri ve işlemciler arasında veri aktarımı büyük bir sorun haline gelmiştir (Oh et al., 2021). Bu sorunu çözmek için insan beyninin çalışma prensibi taklit edilerek yapılan araştırmalara ilgi artmıştır. Beyinden ilham alan (Pedretti et al., 2017; Xinyu Wu et al., 2015; H. Yu et al., 2021) veri işleme, duyusal verilerin gerçek zamanlı işlenmesinde beyin benzeri performans elde etmeyi amaçlayan ve gelişmekte olan bir alandır. Böyle bir hesaplama sistemine ulaşmada bazı sınırlayıcı zorluklar vardır. Bu zorluklar arasında ölçeklenebilir ara bağlantı cihazları, ultra düşük güç tüketimi ve öğrenmeyi donanımda uygulamak için güçlü nöromorfik hesaplama yapıları ile kompakt, büyük ölçüde paralel bir mimari oluşturmak yer almaktadır (Kuzum et al., 2012). Bu zorluklar göz önünde

bulundurularak faz değiştiren malzemeler kullanan cihazlarla simetrik ve asimetrik sinaptik plastisitenin uygulanmasını sağlayan programlama stratejileri, malzeme özellikleri ve spike diyagramları incelenmiş, cihaz çalışma düzeni ve dinamik plan yapısının uyarlanmasıyla enerji tüketiminin optimize edilebildiği bir çalışma sunulmuştur (Kuzum et al., 2012).

Sinaptik elektronikler, beyindeki nöronların birbirleriyle iletişim kurma şeklini taklit eden, bilgiyi insan beynine benzer şekilde işleyebilen, insan beyni kadar verimli ve hataya dayanıklı, ancak daha yoğun bir biçimde elektronik sistemler oluşturmayı amaçlayan yeni bir çalışma alanıdır (Kuzum et al., 2013). İşlem hızı ve enerji tüketimi, makine öğrenimi, derin öğrenme ve veri yoğun uygulamalarda bilgisayarların hesaplama performansı için kritik öneme sahiptir. Bu nedenle, biyolojik beynin hızı ve etkinliği uzun süredir araştırmaların ana odak noktası olmuştur. Bu durum, beynin işleyişine dayanan hesaplama ve sinyal işleme teorileri, formülleri ve tasarımlarının taklit eden çalışmalarla sonuçlanmıştır. Araştırmalar, faz değişimli sinaptik cihazların iki boyutlu çapraz çubuk dizilerinin, beyinden esinlenen bir donanım mimarisinde beyin benzeri öğrenme oluşturmak için kullanılabileceğini ve ilişkisel öğrenme ve örüntü tanıma görevlerine izin verdiğini göstermiştir (Eryilmaz et al., 2013; P. Lin et al., 2020).

Nano ölçekli sinaptik unsurların bireysel cihaz düzeyinde işlev görebileceğine dair deneysel çalışmalardan elde edilen kanıtlara rağmen, ağ düzeyindeki çalışmalar simülasyonlarla sınırlı kalmıştır (Schuman et al., 2017). Deneyler, faz değişimli sinaptik cihazların biyolojide insan beyninin organize edilme şekline benzer şekilde ızgara benzeri bir yapıya bağlanmasıyla dizi düzeyinde ilişkisel öğrenmenin mümkün olduğunu göstermiştir (Eryilmaz et al., 2014). Bu, sistemin farklı cihazlara uyum sağlayabildiğini ve hücre direnci seviyelerindeki büyük farklılıkların eğitim seanslarının miktarının artırılmasıyla dengelenebileceğini göstermiştir.

Vektör matris çarpımı (Amirsoleimani et al., 2020), derin öğrenme uygulamalarında zaman ve enerji maliyetleri açısından büyük bir sorun teşkil etmektedir. Memristör tabanlı vektör matris çarpımının güç tüketimi üzerine yapılan araştırmalar bazı sorunları ortaya çıkarmıştır (Hu et al., 2017; Shafiee et al., 2016). Bayat ve arkadaşları, memristör dedektörü ile donatılmış bir sınıflandırıcı üzerinde bir çalışma yürütmüştür (Bayat vd., 2017). Knowm şirketi, yapay zeka ve derin öğrenme uygulamaları için çok kullanışlı olan ikili anahtarlama ile hem anti-Hebbian hem de Hebbian kurallarını kullanan bir kategorize edici ürün piyasaya sürmüştür (*Anti-Hebbian and Hebbian (AHaH) Plasticity*, 2017). Derin öğrenme uygulamalarında, memristör tabanlı hesaplama süreçleri kullanılırken, özellikle çip üzerinde öğrenme sırasında, sinaptik ağırlığın dinamik aralığı nedeniyle ortaya çıkan sınırlamalar ve zorluklar vardır. Mobil

cihazlarda öğrenme bilgisi edinirken Yapay Zekâ (YZ) işlevlerinin zarar görmesini önlemek için veri sıkıştırma veya kırpma teknikleri önerilmiştir (Mao & Dally, 2016).

Deng ve arkadaşları, oluşturdukları bir uygulamada öğrenmenin çeşitli aşamalarındaki enerji tüketiminin bir analizini yapmışlardır (Deng et al., 2016).

Geleneksel donanım platformları, işlemci ve harici bellek arasında veri aktarımı ihtiyacı nedeniyle öğrenme ile ilgili görevleri tamamlamak için büyük miktarda enerji gerektirir (Schuman et al., 2017). Analog ağırlık depolama kullanan beyinden ilham alan cihaz teknolojileri ile bir cihaz geliştirilmiş, algısal görevlerin daha verimli bir şekilde tamamlandığını gösteren bir çalışma yapılmıştır (Yao et al., 2017). Bu cihaz, ağırlığını her iki yönde de sürekli olarak ayarlayabilme özelliğine sahiptir. Deneyler, paralel anlık eğitim ile 1024 hücreli bir dizinin gri ölçekli yüzleri sınıflandırmak için kullanılabileceğini göstermiştir. Çalışma sonucunda daha düşük enerji tüketimi ile hesaplama işlemleri gerçekleştirilmiştir.

Shamsi ve arkadaşları tarafından üç tasarım seviyesinde sunulan bir sütunlu organize bellek (COM) (Shamsi et al., 2018) donanım mimarisi önerilmiştir. Seviye I'de, COM mimarisi ile uyumlu düşük güçlü bir devre tanıtılmıştır. Seviye II'de, önerilen bir nöron düzeneği ve tek bir memristör çapraz çubuk dizisi kullanılarak bir Winner-Take-All (WTA) algoritması (S. Li et al., 2017) modülü uygulanmıştır. Seviye III'te, WTA modülleri ve memristör çapraz çubuk dizileri birleştirilerek COM tabanlı bir donanım mimarisi oluşturulmuştur. COM donanımını eğitmek için ex-situ yöntemi kullanılmış ve uygulamanın tüm tasarım seviyelerinde simülasyonlar gerçekleştirilmiştir. Çalışmanın birincil odak noktası, nöron devresinin elektriksel güç tüketimini değerlendirmektir (Shamsi et al., 2018).

Çalışmalarında (Saxena et al., 2018), NeuSoC sisteminin enerji tüketimini tahmin etmek için analitik modeller ve devre simülasyonlarının bir kombinasyonunu kullanmışlardır. Farklı çalışma koşulları altında sistemin enerji tüketimini değerlendirmişler ve enerji verimliliği üzerinde en büyük etkiye sahip devre ve cihaz parametrelerini belirlemişlerdir. Araştırmacılar ayrıca sistemin enerji tüketimini azaltmak için tasarım teknikleri önermişlerdir. Son olarak, elde ettikleri sonuçları mevcut NeuSoC sistemleriyle karşılaştırmışlar ve önerdikleri tekniklerin enerji verimliliğini %30'a kadar artırabileceğini göstermişlerdir. Ayrıca, CMOS memristör konsepti için bir emülatöre dayalı CMOS sinaps devrelerini, sistemlerin prototipini oluşturmanın bir yolu olarak sunmuşlar ve pratik memristör cihazlarını normal CMOS ile oluşturmuş ve birleştirmişlerdir. Uçucu olmayan bellek (NVM) veya memristif cihazların geliştirilmesi, derin öğrenmenin bir CMOS katmanı üzerinde karışık sinyalli entegre devrelerle

entegre edildiğinde enerji açısından verimli bir şekilde gerçekleştirilebileceğini göstermiştir. Uçucu olmayan Çip Üzerinde Nöromorfik Sistemler (Saxena et al., 2018) hedefine ulaşmak, çeşitli algoritmik zorlukların ele alınmasını ve bu zorluklara uygun enerji verimli çözümler bulunmasını gerektirmektedir.

Sinirbilim (Sejnowski et al., 1988) alanında, popülasyon kodlama teorisi, sinirsel mekanizmaların hatalara karşı dayanıklı hesaplamalar yapabileceği gösterilmiştir (Mizrahi et al., 2018; Sejnowski et al., 1988). Bu teori, nanoelektronik teknolojisiyle birleştirildiğinde, küçük boyutlu, gürültülü ve hata yapmaya yatkın cihazlarla güvenilir hesaplama yapılmasını mümkün kılabilir. Popülasyon kodlama teorisi CMOS teknolojisiyle uygulanabilir olsa da bu sistemler genellikle yüksek alan veya enerji gereksinimlerine sahiptir. Mizrahi ve ekibi, nano ölçekli manyetik tünel bağlantılarının bu gereksinimleri karşılamak için kullanılabileceğini göstermiştir (Mizrahi et al., 2018).

CMOS teknolojisi ile memristörlerin birleştirildiği bir devrede, CMOS bileşenleri her bir kabloya bağlanır. Bu bağlantı, özel bir "CMOL" arayüzü sayesinde gerçekleştirilir ve bu arayüz, devredeki ek çapraz çubuk yapısındaki tekil memristörlere erişim imkanı sağlar (Prezioso et al., 2015). Bir tür hibrit nöromorfik ağ olan CrossNets, nöron hücre gövdelerinin CMOS tabanlı donanım modellerini memristif çapraz çubuklarla entegre etme potansiyelinden faydalanmaktadır. Bu yenilikçi mimaride, çapraz çubuğun telleri aksonlar ve dendritler olarak işlev görürken, memristörler biyolojik sinapsların davranışını taklit eder. Metal oksit memristörlerin basit, iki terminalli, transistörsüz topolojisi, CMOS modelli memristörlere (Pershin & Di Ventra, 2010), kayan kapıya (Hasler & Marr, 2013), 2013) ve ferroelektrik (Kaneko et al., 2014) bellek hücrelerine dayalı olanlar da dahil olmak üzere, CrossNets'in saf CMOS nöromorfik ağlara kıyasla çok daha yüksek yoğunluk elde etmesini sağlayabilmektedir. Prezioso ve ekibi, transistör içermeyen entegre bir çapraz çubuk yapısı kullanarak, bunun yerine metal oksit memristörlere dayanan işlevsel bir sinir ağının deneysel bir gösterimini gerçekleştirmiştir (Prezioso et al., 2015). Ağ, doğrudan kendi operasyonel ortamında eğitime tabi tutuldu, yani harici bir bilgisayar modeline dayanmadı ve Manhattan güncelleme kuralını kullandı (Lim et al., 2019). Manhattan güncelleme kuralı ve toplu mod delta kuralı temelde aynıdır, tek fark donanım uygulamasını kolaylaştırmaya yarayan ikili niceleme kullanımıdır (Prezioso et al., 2015).

Derin analog Yapay Sinir Ağları (YSA'lar) karmaşık sınıflandırma problemlerini çok yüksek bir doğruluk derecesiyle çözebilmektedir (Musisi-Nkambwe et al., 2021). Kuantum bilgisayarlar, uygulamalarda sağladıkları doğruluk avantajlarına rağmen, hesaplamalar için çok

fazla enerji tüketir ve bu da pratik kullanım alanlarını sınırlayabilir (Lukoševičius & Jaeger, 2009). Bu durum, uygulamalarda elde edilen doğruluk avantajlarının kullanışlılığını gölgelemektedir. Wijesinghe ve arkadaşları, nano ölçekli dirençli cihazların rastgeleliğinin bir spike nöronun davranışını taklit etmek için nasıl kullanılabileceğini ve daha sonra bu durumu ele almak için derin stokastik SNN'lere nasıl dahil edilebileceğini önermişlerdir. Algoritmik olarak, bir Yapay Sinir Ağını (Sharma et al., 2020) bir Spiking Sinir Ağına (Lashkare et al., 2018) dönüştürmek için eğitim sürecinin nasıl değiştirilebileceğini açıklarken, bu cihazların sunduğu rastgele aktivasyon işlevini kullanmaya devam ederler. Sinaptik ağırlıklarla aynı işlevleri yerine getirmek için stokastik anımsatıcı nöronları anımsatıcı çapraz çubuklarla birleştiren devre mimarileri tasarladıkları bir çalışma sunmuşlardır (Wijesinghe et al., 2018). Bu çalışmada, tasarlanan sistemin benzerlerine kıyasla enerji tüketimi açısından daha verimli olduğunu kanıtlamışlardır.

Nöromorfik hesaplama sistemleri için bir donanım ağı yapısının oluşturulması temelde bellek dizilerinin entegrasyonudur. Günümüzde çapraz çubuk dizisi tabanlı bellek sistemlerinin parazit akımı göz önüne alındığında, donanım tabanlı sinir ağlarının parazit önleme yeteneği son derece zayıf olabilir (Lim et al., 2019). Bu, geliştirilen sistemin uygulanmasında önemli bir zorluktur ve olası okuma hatalarına ve eğitim sürecinde artan enerji tüketimine yol açabilir. Bu durumu çözmeye yönelik çalışmalar gerçekleştirilmiştir (Gul, 2019).

Elektronik sinaps cihazlarının kullanıldığı donanımlar için öğrenme algoritması çalışmaları gerçekleştirilmiştir. Bir çalışmada, sınırlı ve süreksiz iletkenlik özelliklerine sahip elektronik aletler kullanan donanım tabanlı bir derin sinir ağı için geri yayılım algoritması tabanlı bir öğrenme kuralı önerilmiştir (Lim et al., 2019). Bu algoritma, donanım tabanlı ağırlık ayarlamalarının yanı sıra hem ileri hem de geri yayılmayı sağlayan esnek bir öğrenme mekanizması içermektedir. Ayrıca, bu algoritma enerji verimli ve yüksek hızlı derin sinir ağlarının yürütülmesine yardımcı olacak şekilde uyarlanmıştır. Bu çalışmada, elektronik sinaps cihazlarının öğrenme performansı, üç katmanlı bir sensör ağı kullanılarak yapılan simülasyonlarda çeşitli iletkenlik yanıtlarına ve ağırlık güncelleme yöntemlerine göre değerlendirilmiştir.

Spike Sinir Ağları (SNN'ler), biyolojik beynin bilgiyi hızlı ve doğru bir şekilde işleme yeteneğinden ilham alır ve biyolojik sinir ağlarının sinirsel kodlarını, dinamiklerini ve devrelerini kopyalamayı amaçlar. SNN'ler, bellek içi hesaplama kullanarak denetimsiz öğrenme uygulaması için büyük bir potansiyele sahiptir (Sourikopoulos et al., 2017). Bu durum göz önüne alındığında, uçucu olmayan bellek (eNVM) cihazlarında Spike Sinir Ağları (SNN'ler) ile

öğrenmenin enerji verimliliğini artıran algoritmik bir optimizasyon sunmaktadır (Y. Shi et al., 2019). Bir başka çalışmada, bellek ile merkezi işlem birimi arasındaki veri aktarım gereksinimini en aza indirmek için, bellek içi hesaplama mimarileri de dikkate alınarak, STT-MRAM ve SRAM kullanılarak bir YSA model sisteminin fiziksel uygulaması için kapsamlı bir çalışma yapılmıştır.

Beyinden ilham alan paralel hesaplama, insan beyninin bilgiyi işleme şekline çok benzer şekilde, verileri paralel olarak işlemek için birbirine bağlı işlemcilerden oluşan bir ağ kullanma fikrine dayanmaktadır (B. Gao et al., 2016). Bu tür bilgi işlem, büyük miktarda veriyi çok daha kısa sürede işleyebildiği için geleneksel bilgi işlem yöntemlerinden daha verimli olacak şekilde tasarlanmıştır. Ayrıca, geleneksel bilgi işlem yöntemleriyle aynı miktarda güç gerektirmediği için daha enerji verimli olma potansiyeline de sahiptir. Doğrusal olmayan ve asimetrik iletkenlik-güncelleme özelliklerine sahip yapay sinapsların faydalarına rağmen, bir donanım yapay sinir ağı, bir yazılım yapay sinir ağının eğitim ve çıkarım doğruluğuyla eşleşemez. Bu durum için doğrusal ve simetrik iletkenlik-güncelleme özelliklerine sahip yeni bir yapay van der-Waals hibrit sinaps geliştirilmiştir (S. Seo et al., 2020). Bu çalışmada, iletkenliği seçici olarak artırmak ve azaltmak için tungsten diselenid (WSe_2) ve molibden disülfür (MoS_2) kanalları kullanılmıştır. Daha sonra, bir donanım yapay sinir ağı için hibrit bir sinapsın potansiyeli, eğitim ve çıkarım simülasyonu yoluyla gösterilmiştir.

Rahimi Azghadi ve çalışma arkadaşları tarafından yürütülen bir çalışma (Rahimi Azghadi et al., 2020) ile CMOS, SiO_x tabanlı memristif ve karma CMOS-memristif teknolojiler kullanılarak nöromorfik hesaplama üzerine kapsamlı bir inceleme gerçekleştirilmiştir. Bu çalışma, biyolojik nöronların ve sinapsların bazı yönlerini taklit edebilen tasarımları içermektedir. Araştırma, nöronları taklit edebilen bileşenlerin model sınıflandırmaları veya görüntü tanıma gibi görevlerde kullanılabileceğini göstermiştir. Cihaz çeşitliliklerine ve ideal olmayan koşullara rağmen, geliştirilen unsurların iyi performans gösterdiği gözlemlenmiştir.

Beyinden ilham alan sinaptik nano-elektronik cihazlar, düşük enerji tüketimi ve veriyi paralel olarak işleyebilme yetenekleri gibi biyolojik nöronlara benzer özellikleri nedeniyle giderek daha fazla popülerlik kazanmaktadır. Metal oksitten yapılan dirençli anahtarlamalı bellek cihazları, düşük maliyetli, üretimi kolay ve tamamlayıcı metal oksit yarı iletken (CMOS) teknolojisi ile uyumlu olmaları nedeniyle sinaps oluşturmak için son derece arzu edilmektedir (B. Gao et al., 2016). Bu nedenle, bu alandaki bir çalışmada, nöromorfik uygulamalar için basit, tek katmanlı ve nano ölçekli memristör tabanlı yapay bir sinaptik cihaz sunulmuştur (Gul, 2020).

Başka bir çalışmada (Shymkovych et al., 2021), Radyal Tabanlı Fonksiyon (RBF) sinir ağlarının Gauss aktivasyon yöntemini Programlanabilir Kapı Dizileri (FPGAs) donanımında uygulamayı amaçlayan bir araştırma önerisi sunulmuştur.

Günümüzde, Von Neumann darboğazını aşmak amacıyla bellek içi hesaplamada sinaptik cihazların kullanımı konusunda önemli ilerlemeler kaydedilmiştir (Zanotti et al., 2020). Ancak, derin sinir ağlarının donanımda verimli bir şekilde uygulanabilmesi için doğrusal olmayan aktivasyon işlevlerini yerine getirebilen kompakt nano cihazlara ihtiyaç duyulmaktadır (Haoxiang & S, 2021). Bu doğrultuda, vanadyum dioksit (VO_2) tabanlı bir Mott aktivasyon nöronu ile bir iletken köprü rasgele erişimli bellek (CBRAM) çapraz bağlantı dizisinin entegre edildiği başarılı bir çalışma gerçekleştirilmiştir (Oh et al., 2021). Bu araştırmada, Mott aktivasyon nöronu analog alanda düzeltilmiş doğrusal birim (ReLU) işlevini kullanmaktadır. Nöron cihazları, analog tamamlayıcı metal-oksit yarı iletken uygulamalara göre önemli ölçüde daha az enerji tüketmekte ve daha az alan kaplamaktadır. Mott aktivasyon nöronları ile çalışan LeNet-5 ağı, MNIST veri kümesinde ideal yazılım doğruluğuna yakın bir doğruluk oranı olan %98,38'e ulaşmıştır. Ayrıca, bu çalışmada (Oh ve diğerleri, 2021), Mott aktivasyon nöronları ve CBRAM çapraz bağlantı dizilerinin kombinasyonu kullanılarak büyük ölçekli görüntü kenar tespiti süreçleri de gerçekleştirilmiştir.

İki boyutlu (2D) geçiş metali kalkojenitler (TMC'ler) ve bunların heteroyapıları, beyinden ilham alan nöromorfik hesaplama sistemlerinde, ileriye dönük bellek ve sinaptik cihazlar için önemli yapı taşlarıdır ve çeşitli elektronik ve optoelektronik cihazlarda kullanım potansiyeline sahiptir (Kwon et al., 2022). Bu çalışma, nöromorfik hesaplama uygulamalarında kullanılan iki boyutlu geçiş metali kalkojenitlere (2D TMC'ler) dayalı yüksek performanslı memristörlere dair kapsamlı bir inceleme sunmaktadır. Çalışmada iki boyutlu geçiş metali kalkojenit malzemeler ve heteroyapılar ele alınarak memristif cihazların mevcut durumu detaylı bir şekilde değerlendirilmektedir (Kwon et al., 2022). Bu araştırmanın amacı, iki boyutlu geçiş metali karbürlerden üretilen nöromorfik memristörlerin üretimi ve karakterizasyonuna genel bir bakış sunmak ve bu malzemelerin ve cihazların gelecekte karşılaşabileceği zorlukları ve potansiyel fırsatları tartışmaktır.

Memristif cihazlar, düşük enerji tüketimi, ölçeklenebilirlik ve direnç değişiminin çok seviyeli doğası (plastisite) gibi özellikleri sayesinde derin öğrenme ve yapay zekâ uygulamalarında enerji tüketimi ve ölçeklenebilirlik gibi sorunları çözmede oldukça umut vaat etmektedir (Kwon et al., 2022). Çok seviyeli plastisite, memristörlerin fiziksel nöromorfik hesaplama sistemlerinde sinapsları taklit etmesine olanak tanır (Shvetsov et al., 2022). Shvetsov

ve çalışma arkadaşları, çapraz bağlantı geometrisinde üretilen Cu/poly-p-xylene (PPX)/Au hafıza elemanlarını inceleyen bir çalışma gerçekleştirmiştir. Yapılan çalışmalar sonucunda, tek bir memristörün döngüden döngüye geçişi ve birkaç memristörün cihazdan cihaza geçişi üzerine yapılan deneylerde direnç değişim voltajlarının yüksek tekrarlanabilirliği gösterilmiştir. Elde edilen memristörlere dayanarak, basit desenleri sınıflandırmak üzere eğitilebilen bir donanım nöromorfik ağı oluşturulmuştur.

Tablo 3, CMOS nöronları ve memristör tabanlı nöronlar için güç tüketimi, nöron başına enerji tüketimi değerlerini sunmaktadır. Bu makalede, memristör düğümü ile birlikte CMOS devreleri ve transistörlü bir memristör düğümünün kombinasyonu tanıtılmaktadır.

Tablo 3. CMOS nöronları ile memristör tabanlı nöronların karşılaştırılması

Referans	**Cihaz Tip/Model**	**Konfigürasyon**	**Enerji/Spike**
(Indiveri, 2003)	CMOS	18–20 transistör	0.3–1.5 μW, 2850 pJ/spike
(Y. J. Lee et al., 2004)	CMOS	90 transistör	163.4 μW
(Cruz-Albrecht et al., 2012)	CMOS	16 transistör	40.2 pW, 0.4 pJ/spike
(Sourikopoulos et al., 2017)	CMOS	9 transistör	4 fJ/spike
(Shamsi et al., 2018)	CMOS	14 transistör	4.3 pJ/spike
(Wijekoon & Dudek, 2008)	CMOS	14 transistör	8–40 μW
(Babacan et al., 2016)	CMOS	1 transistör emülatör + 3 transistör	60–110 μW
(Saxena et al., 2018)	CMOS + Memristör	Memristor emülatör (8 transistor)	14 fJ–1.4 pJ/spike
(Mizrahi et al., 2018)	CMOS + Memristör	1 memristör + 1 manyetik coupling + CMOS devre	3.3 μW, 150 pJ/junction
(Z. Wang et al., 2018)	Memristör	~6 x 4	~5 (nJ/spike)
(X. Zhang et al., 2018)	Memristör	5 x 5	~700 (nJ/spike)
(Lu et al., 2020)	Memristör	<10 × 10	16 (fJ/spike)
(Feali, 2021)	Memristör	<10 × 10	16 (pJ/spike)

Yapılan çalışmalar ve bu çalışmalarda kullanılan model, alan ve enerji tüketimi gibi özellikleri Tablo 3'te gösterilmiştir. Bu tabloda, kullanılan memristör sayısı ile enerji tüketimi arasındaki dengeye dikkat etmek önemlidir. Bu durumun belirlenmesi uygulamalara göre değişiklik gösterebilir. Örneğin, Tablo 3 karşılaştırıldığında, 14 transistör kullanan Wijekoon

Dudek modeli, Babacan'ın yaklaşımına benzer tüm yükselme ve patlama türlerini gerçekleştirebilmekte ancak enerji tüketimi açısından ele alındığında, Babacan'ın memristör nöronuna göre %40 daha az güç kullanmaktadır. Tablo 3'te enerji tüketim miktarlarına bakıldığında en umut verici nöron Sourikopoulos'un çalışması olarak görülmektedir. Ancak yapay zekâ uygulamalarının en verimli formu yalnızca tek bir duruma dayanılarak değerlendirilemez. Donanım mimarisi tasarımında enerji tüketimini azaltmak için çipin hızı ve boyutu göz önünde bulundurulmalıdır. Bununla birlikte, bu koşulların en ideal durumu için belirli bir çip tasarımında bazı tavizler verilmesi gerekebilmektedir.

Memristör tabanlı donanım hızlandırıcılarının güvenilirlik üzerindeki etkisi

Memristör teknolojisi, makine öğrenimi, yapay zekâ, doğal dil işleme ve görüntü işleme gibi geniş bir uygulama yelpazesi için büyük bir potansiyel taşımaktadır. Ancak, bu ürünlerin ticari hale gelmesi için güvenilirlik, üretim, tutarlılık ve ölçeklenebilirlik gibi bazı önemli zorlukların ele alınması ve aşılması gerekmektedir (Mehonic et al., 2020). Memristör teknolojileri ailesi çeşitlilik göstermekte olup, farklı teknolojiler farklı cihaz ve sistem kusurlarıyla birlikte gelmektedir. Bu zorlukların yanı sıra, memristör tabanlı donanım hızlandırıcılarının güvenilirliği ve zorlukları, memristörlerin görece yeni bir teknoloji olması nedeniyle araştırmacılar için aktif ve popüler bir araştırma konusudur.

Memristör teknolojisindeki güvenilirliği etkileyen en önemli zorluklardan biri, olgunlaşmamış üretim süreçlerinden kaynaklanan donanım kusurlarıdır (Kannan et al., 2015) (C.-Y. Chen et al., 2015) (S. Jin et al., 2020). Parametre varyasyonları, yüksek hassasiyetli memristör programlamasıyla kalibre edilebilse de, donanım kusurları geri dönüşümsüzdür ve düzeltilemez (Merced-Grafals et al., 2016).

Memristör teknolojisindeki bir diğer ideal dışı durum ise takılı hata olarak bilinen stuck-at-fault (SAF) ve cihazdan cihaza (D2D) varyasyonlardır (Joksas et al., 2020). Bir arıza durumunda stuck-at-fault (SAF), bir cihazın belirli bir durumda sıkışıp kalmasıyla ilgili önemli bir güvenilirlik sorunudur (M. Liu et al., 2019). Bu durum, yapay sinir ağlarında yanlış ağırlıkların oluşmasına ve hatalı hesaplamalara yol açabilir. D2D varyasyonlarıda, farklı cihazların programlama darbelerine farklı tepkiler vermesi nedeniyle memristörlerin yanlış programlanmasına neden olabilir. Sonuç olarak, memristörler yanlış iletkenlik seviyesine programlanabilir. Bu sorunları ele almak için, okuma ve doğrulama programlama şemaları veya yüksek hassasiyetli işlem birimlerinin çeşitli teknikler önerilmektedir (Le Gallo et al., 2018; Shim et al., 2020).

Bir diğer sorun, birçok memristif cihazda gözlemlenen akım-gerilim karakteristiklerindeki doğrusal olmayanlıktır. Çıkış akımı ile uygulanan voltaj arasındaki doğrusal ilişkinin olmaması ve varsayılan doğrusal ilişkinin *(I = GV)* tüm voltaj aralığında kullanılamaması, memristör çapraz barları kullanarak doğru vektör matris hesaplamalarını engellemektedir (Mehonic et al., 2020).

Alan çalışmaları ile nöromorfik sistemin genel değerlendirilmesi

Literatür incelemesinde ilgili çalışmalarda görüldüğü üzere, bilgisayar sistemlerinde yazılım ve donanım alanında sürekli ve çok hızlı bir gelişim süreci yaşanmaktadır. Bu ilerleme ile birlikte, yapılandırılmış sorunları çözme konusunda başarılı olan Von Neumann mimarisine dayanan geleneksel bilgisayar sistemlerinin, büyük miktarda yapılandırılmamış veriyi işlemek için yetersiz olduğu anlaşılmıştır. Bu durumun sonucunda, son yıllarda daha fazla araştırmacı memristörlerin özelliklerini incelemeye ve daha gerçekçi memristör modelleri oluşturmaya başlamıştır. Memristör tabanlı nöromorfik hesaplama, donanım birimleri, uygulama modelleri, algoritmalar ve entegrasyon teknolojilerini kapsayan farklı disiplinlerin araştırma ve uygulama geliştirme alanı haline gelmiştir. Bu durum, gerçekleştirilen çalışmalarda yeni fırsatlar sunarken, aynı zamanda bazı zorlukları beraberinde getirmektedir. Bu, donanım tabanlı çalışmaların istenen seviyeye ulaşabilmesi için aşağıda sıralanan durumların göz önünde bulundurulması gerektiği düşünülmektedir. Memristör tabanlı nöromorfik donanımın inşasında farklı malzemeler kullanılmalıdır. Tasarlanan memristörler test edilmeli ve analiz edilmelidir. Bu analizler doğrultusunda, daha kararlı ve güvenilir memristör malzemeleri ve üretim yöntemleri izlenmelidir. Birçok memristör tabanlı uygulama düşük doğruluk sorunları yaşamaktadır. Bu sorunun çözümü, memristörlerin güvenilirliğini artırmakta yatmaktadır. Tasarlanan memristör birimlerinin güvenilirliğini artırmanın en önemli yöntemi, yeni üretim malzemelerinin kullanılmasıdır.

Analog veya dijital hesaplamalarda, farklı cihazlardaki memristör cihaz özelliklerindeki ani değişimler, gerçekleştirilen işlemlerin doğruluğunu önemli ölçüde etkileyebilir. Özellikle yüksek doğruluğun çok önemli olduğu bilimsel hesaplamalarda, memristör cihazlarının uygunluk gereksinimi görece yüksektir. Doğrulama yöntemlerinin veya yedeklilik tasarımının kullanılması, bu hataların toleransını bir dereceye kadar artırabilir; ancak bunun sonucunda ek enerji tüketimi ve gecikme meydana gelecektir. Sonuç olarak, memristör tabanlı bellek içi hesaplamaların doğal avantajları zayıflayacaktır. Bu nedenle, geliştirilen uygulamalarda kullanılan memristör cihazlarının uygunluğu önemli bir sorundur.

Literatür taramasından çıkarılan bir diğer sonuç ise, memristör tabanlı yapıların dikkate alınmasıyla, bu özelliklere uygun olarak tasarlanacak yeni öğrenme algoritmaları ile geliştirilen sistemden daha iyi verim elde edilebilir. Ayrıca, yeni nesil bilgisayarların donanım mimarisi, Von Neumann mimarisinden farklı şekilde tasarlanmalıdır. Biyolojik ilham alan bir hesaplama paradigması olarak nöromorfik hesaplama, yapay zekanın ve derin öğrenme süreçlerinin hızlandırılması ile enerji verimliliği açısından optimal çözümler için büyük bir potansiyele sahiptir. Akademi ve sanayide artan araştırma girişimleri ile yakın gelecekte daha güvenilir öğrenme algoritmaları ve daha verimli uygulamalar konusunda iyileştirmelerin gerçekleşmesi beklenmektedir.

MATERYAL VE METOT

Yöntem

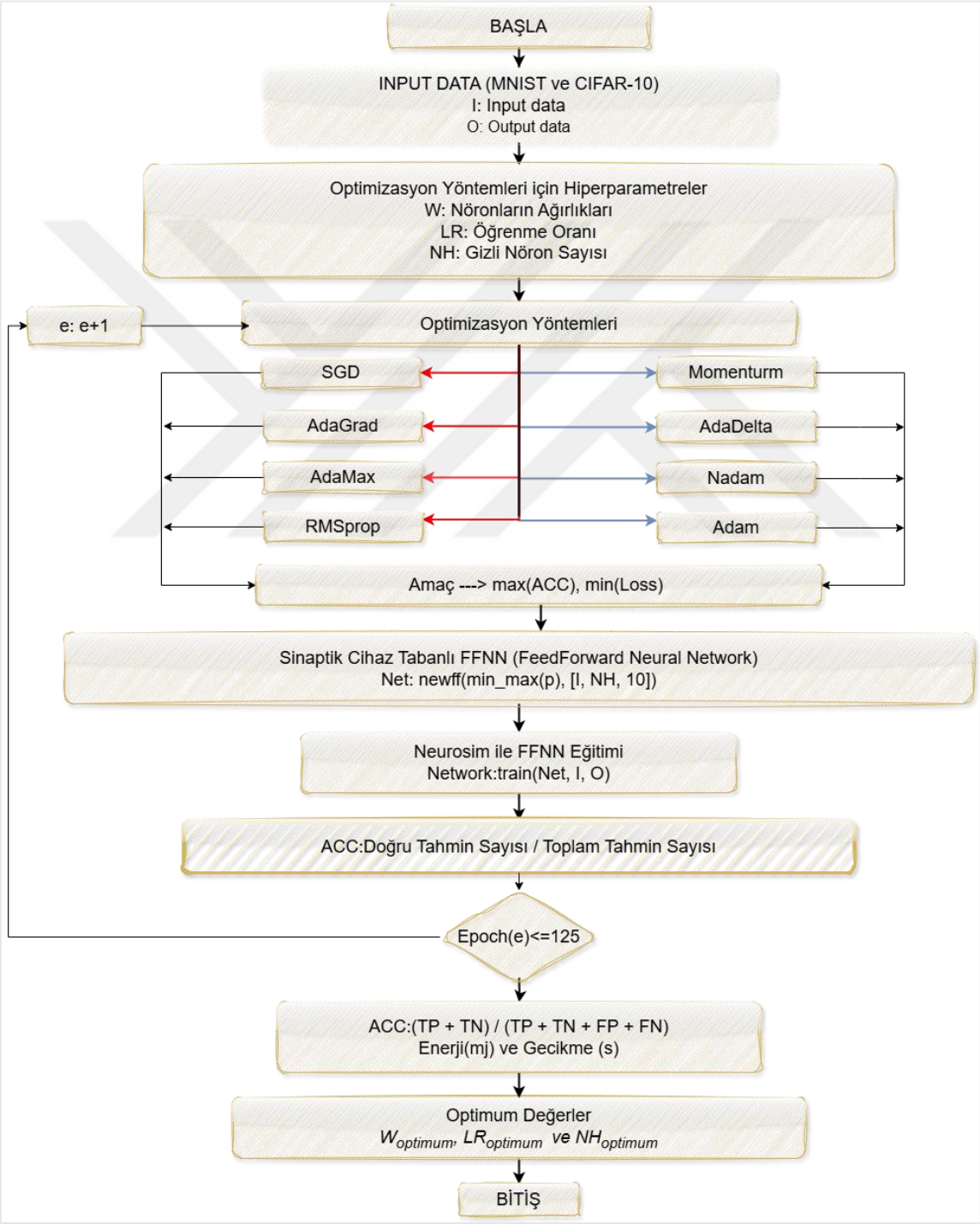
Bu tez çalışmasında, donanım tabanlı bir öğrenme modeli oluşturma sürecinde kullanılan metodolojinin temel bileşenleri kapsamlı bir biçimde incelenmekte ve oluşturulacak modelin performansı bilgisayar ortamında simüle edilmektedir. Bu metodoloji, optimizasyon yöntemlerinin entegrasyonu yoluyla modelin performansını artırmayı hedeflemekte ve çeşitli adımlar içermektedir. İlk olarak, optimizasyon sürecinde kullanılan matematiksel yaklaşımlar tanıtılmakta, ardından, donanım üzerinde öğrenme modelini kurma aşamasında bu yöntemlerin nasıl uygulandığı ele alınmaktadır. Bu çerçevede, her bir adımın modelin genel verimliliğine katkıları ve donanımın sınırlamaları dikkate alınarak yapılan düzenlemeler de açıklanmaktadır.

Bu çalışma, sinir ağlarının donanım tabanlı uygulanabilirliğini inceleyerek, özellikle memristör tabanlı nano-sinaptik cihazların kullanım potansiyelini detaylı bir şekilde ele almaktadır. Memristör gibi gelişmiş malzemelerle üretilen nano-sinaptik cihazlar, sinir ağlarının hem doğruluk hem de işlem verimliliği açısından performansını iyileştirebilmekte, bununla birlikte enerji tüketimini azaltarak geniş çaplı hesaplama gereksinimlerine yönelik etkin çözümler sunabilmektedir. Çalışmada önerilen ve bilgisayar ortamında simüle edilen donanım tabanlı yapay sinir ağı modeli, çevrimiçi öğrenme ve çevrimdışı sınıflandırma görevleri optimum hale getirilmiş olup, özellikle veri yoğun sınıflandırma problemlerinde işlevsel ve ölçeklenebilir bir yapı sunmaktadır.

Donanım tabanlı uygulama süreci, makine öğrenimi ve derin öğrenme algoritmalarının ileri besleme (FeedForward, FF) ve geri yayılım (BackPropagation, BP) gibi temel tekniklerini kapsayarak, FF algoritmasında giriş verisinin ağırlıklı toplamlar ve aktivasyon fonksiyonları üzerinden katmanlar boyunca çıkış katmanına ulaştırılması prensibine dayanmaktadır. Bu katman bazlı iletim süreci, sinir ağı çıktısının belirli bir etiket ile karşılaştırılmasını sağlar, böylece modelin tahmin hatası hesaplanır. BP aşamasında ise, hesaplanan tahmin hatası, modelin performansını iyileştirmek üzere optimizasyon algoritmaları aracılığıyla geri yayılır ve her bir veri noktasına ilişkin ağırlık değerleri anında güncellenerek hata minimize edilir.

Özellikle donanım seviyesinde optimize edilmiş bu sinir ağı modeli, geleneksel gradyan inişi gibi yazılım tabanlı yöntemlerden ayrışarak enerji verimliliği, işlem hızı ve doğruluk açısından ciddi avantajlar sağlamaktadır. Hinduja ve arkadaşları ile Lillicrap ve arkadaşlarının çalışmaları, donanım tabanlı sinir ağlarının hem enerji tasarrufu sağladığını hem de işlem hızını

artırdığını göstermektedir. Bu ağlar, büyük veri kümeleriyle çalışırken gerekli olan hesaplama gücünü minimize etme konusunda da etkili bir çözüm sunmaktadır (Hinduja et al., 2019; Lillicrap et al., 2020). Şekil 18, çalışmanın genel yapısını göstermekle birlikte optimizasyon algoritmalarının makine öğrenimi model performansı üzerindeki etkilerini inceleyen bir akış diyagramını da ifade etmektedir. Şekil 18'in genel akışı, giriş verilerinin işlenmesinden nihai sonuçların değerlendirilmesine kadar olan süreci kapsamaktadır.



Şekil 18. Çalışmada kullanılan metodolojinin akış diyagramı.

Aşağıda Şekil 18'da gösterilen akış diyagramının özet biçiminde açıklaması verilmiş olup ilerleyen bölümlerde bu kısımlar detaylı bir şekilde anlatılacaktır.

- Başlangıç ve Giriş Verisi (Input Data): İlk adımda, MNIST, CIFAR-10 ve Fisher's Iris gibi veri kümeleri bilgisayar ortamında simüle edilen modele girdi olarak verilmektedir. Bu veri kümeleri, el yazısı rakamlar ve küçük renkli görüntüler gibi standart makine öğrenimi test veri setlerini içerir.
- Optimizasyon Yöntemleri İçin Hiperparametreler: Modelin başarısını etkileyen öğrenme oranı, nöron ağırlıkları ve gizli katman sayısı gibi çeşitli hiperparametreler belirlenir. Bu ayarlar, modelin performansını optimize etmek için detaylı ayarlamalar yapılması gereken kritik parametrelerdir.
- Optimizasyon Yöntemleri: Akış diyagramında Stokastik Gradyan İnişi (SGD), AdaGrad, AdaMax, RMSProp, Momentum, AdaDelta, Nadam ve Adam gibi çeşitli optimizasyon algoritmaları listelenmiştir. Bu algoritmalar, modelin öğrenme sürecinde ağırlıkların güncellenme şeklini belirler.
- Hedefler (Amaç): Optimizasyon sürecinin amacı, doğruluğu (ACC) maksimize ederken kaybı (Loss) minimize etmektir. Bu iki metrik, model performansını değerlendirmede kritik rol oynar.
- Sinaptik Cihaz Tabanlı FFNN: Optimizasyon süreci, bir İleri Beslemeli Yapay Sinir Ağı (FFNN) modelini eğitmek için kullanılır. Sinaptik cihazlar terimi, modelin donanım veya yazılım uygulamalarında nöromorfik sistemleri temsil ediyor olabilir.
- Neurosim ile Eğitim: YSA'nın eğitimi, Neurosim adlı bir simülasyon aracı kullanılarak gerçekleştirilir. Bu araç, nöromorfik hesaplamayı simüle ederek sinir ağı performansını değerlendirmeye yardımcı olmaktadır.
- Doğruluk ve Optimum Değerlerin Hesaplanması: Modelin doğruluk oranı (ACC), doğru tahmin sayısının toplam tahmin sayısına oranı olarak hesaplanır. Bu adımda ayrıca modelin optimum parametre değerleri belirlenir.
- Sonuçlar: Nihai sonuçlar ve değerlendirme metrikleri sunulmakta, akış diyagramı "BİTİŞ" adımıyla tamamlanmaktadır.

Şekil 18, optimizasyon algoritmalarının karşılaştırmalı performans analizine yönelik bir süreç akışını sunmakta olup, makine öğrenimi alanında hiperparametre ayarlarının ve optimizasyon yöntemlerinin etkisini incelemek için tasarlanmıştır. Burada optimizasyon yöntemleri

kullanılarak donanım tabanlı bir öğrenme modeli oluşturmak için kullanılan metodolojinin temel yönleri gösterilmektedir.

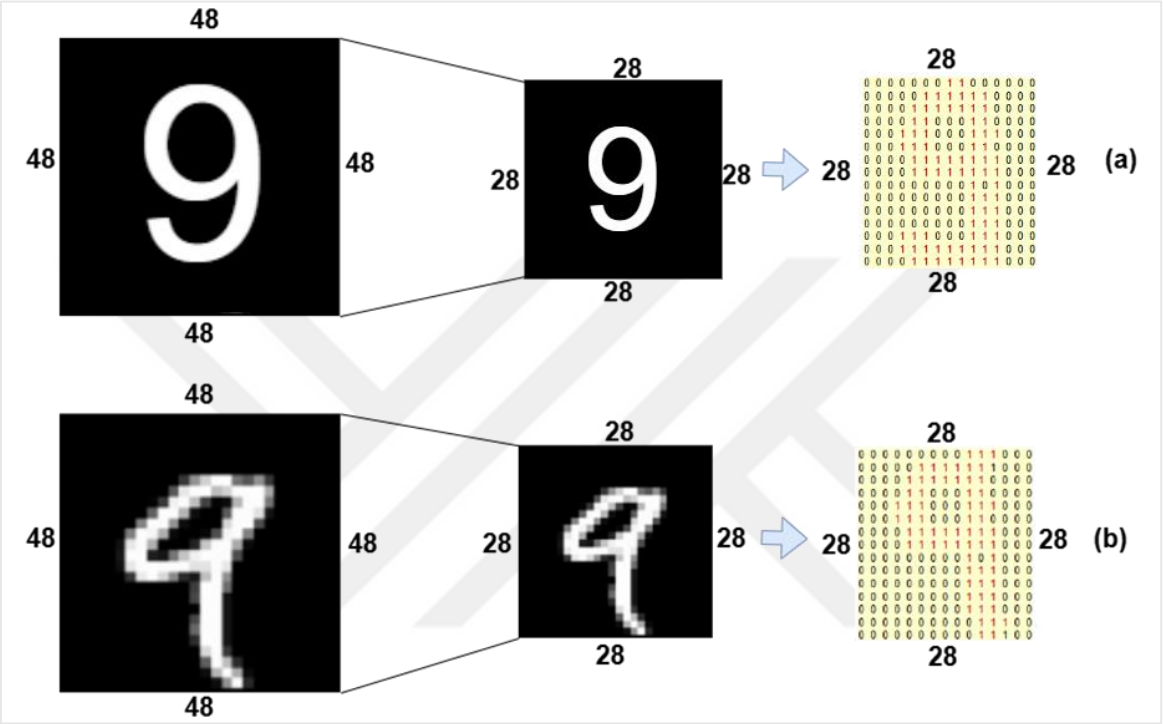
Tablo 4. Uygulamanın genel çalışma mantığını anlatan sözde kod

```
Input: InputData, Hyperparameters, Optimizers
Output: BestModel, BestAccuracy, BestLoss
Begin
  Initialize Variables
      data ← LoadData (InputData)
      params ← SetHyperparameters (Hyperparameters)
      bestModel ← NULL
      bestAccuracy ← 0
      bestLoss ← ∞
  For each optimizer in Optimizers do
      Initialize Model with Current Optimizer
          model ← InitializeModel (params, optimizer)
      Randomly Initialize Model Weights
          RandomInitializeWeights(model)
      Train the Model using Data
          Train (model, data)
      Evaluate Model Performance
          accuracy ← CalculateAccuracy (model, data)
          loss ← CalculateLoss (model, data)
      Compare and Update Best Model if Necessary
          if accuracy > bestAccuracy or (accuracy == bestAccuracy and loss < bestLoss) then
              bestModel ← model
              bestAccuracy ← accuracy
              bestLoss ← loss
          End if
  End For
  Return Results
      return bestModel, bestAccuracy, bestLoss
End
```

Veri Seti Tanımlaması ve Kullanılan Veri Setleri

El yazısı karakter tanıma amacıyla MNIST veri kümesi (Li Deng, 2012) kullanılmıştır. MNIST, yaygın olarak el yazısı rakamların tanınması üzerine yapılan çalışmalar için standart bir test veri kümesi olarak kullanılmaktadır. Ayrıca bilgisayarlı görü algoritmalarının performansını değerlendirmede oldukça etkin bir ölçüt sunar. Bu tez çalışmasında, bilgisayar tarafından yazılmış karakter tanıma sürecini incelemek amacıyla her bir rakamın dijital görüntüsünü bilgisayar ortamında üreterek kendi veri kümemizi oluşturduk. Her bir karakter görüntüsü, veri ön işleme aşamasında sayısallaştırılarak başlangıçtaki 48×48 piksel çözünürlükten 28×28 piksele normalize edilmiştir. Bu, sinir ağımızın giriş katmanındaki

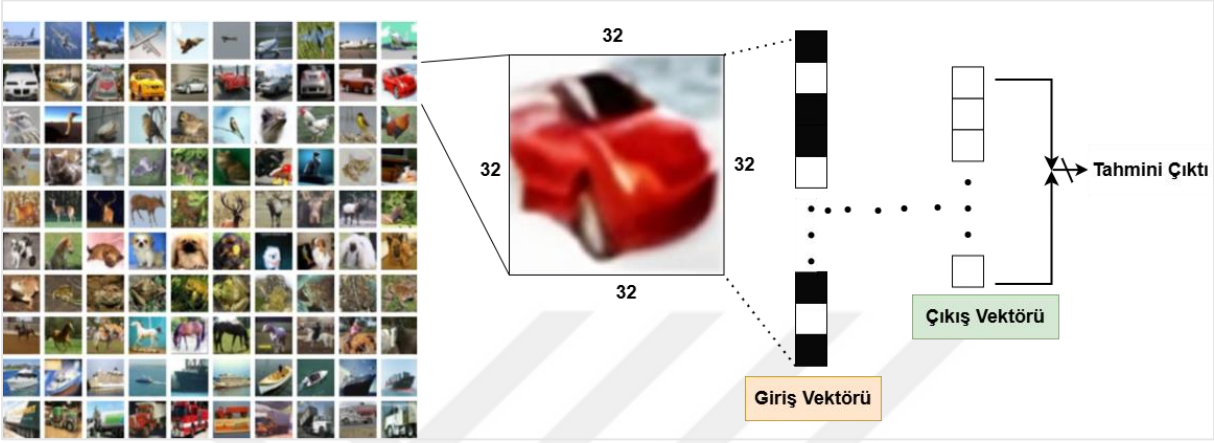
nöronlar için gerekli olan giriş vektörlerini dikey ve yatay özellik vektörlerinin birleşimi şeklinde sunmamıza olanak sağlamaktadır. Neticede, önerilen sinir ağı modelimiz, bilgisayar ortamında yazılmış bir karakter ile el yazısı bir karakter için toplamda 56 benzersiz özellik taşıyan bir giriş vektörünü işlemektedir. Bilgisayar tarafından üretilen karakterlerden oluşan yeni bir veri kümesi, metin görüntülerinin 28 × 28 piksele sayısallaştırılmasıyla oluşturulmuş olup ve Şekil 19 (a) ve Şekil 19 (b)'de gösterilmektedir.



Şekil 19. (a) Bilgisayar ile yazılmış bir rakamın uygulama ile binary hale getirilmesi **(b)** Elle yazılmış bir rakamın uygulama ile binary hale getirilmesi.

Ayrıca bu tez çalışmasında, donanım tabanlı bir sinir ağı mimarisinin performansını değerlendirmek amacıyla CIFAR-10 veri kümesinden (McCrary, 1992) faydalanılmıştır. CIFAR-10, bilgisayarla görme alanında sıklıkla kullanılan, geniş kapsamlı ve karmaşık bir görüntü sınıflandırma veri setidir. Bu veri seti, her biri 32×32 piksel boyutunda ve renkli olan 60.000 görüntüden oluşmaktadır ve on farklı sınıfa dağıtılmıştır. Bu sınıflar, günlük hayatta karşılaşılan çeşitli nesne ve hayvan kategorilerini içermektedir. CIFAR-10 veri kümesindeki her görüntü, önceden belirlenmiş on sınıftan birine atanmıştır, bu da veri kümesini denetimli öğrenme tabanlı görüntü sınıflandırma çalışmaları için ideal hale getirmektedir. Veri kümesinde bulunan çeşitli nesneler ve karmaşık arka planlar, sınıflandırma modelleri için oldukça zorlayıcı bir ortam sunmakta, böylece algoritmaların sınıfları ayırt edebilmek için derin özellikleri öğrenmesini zorunlu kılmaktadır (McCrary, 1992). Tablo 5'te sözde kodu yazılan ptyhon programlama dilinde yazılan kod ile CIFAR-10 veri kümesini işleyip eğitim ve test verilerini

belirli dosyalara kaydeder. Öncelikle, veriler belirtilen dizinlerdeki dosyalardan yüklenir ve ardından eğitim verileri birleştirilerek *patch_train.txt* ve *label_train.txt* dosyalarına kaydedilir. Test verileri de benzer şekilde *patch_test.txt* ve *label_test.txt* dosyalarına kaydedilir. Algoritma, eğitim ve test verilerini ayrı dosyalara kaydederek veriyi model eğitimi için kullanılabilir hale getirir.



Şekil 20. Çalışmada CIFAR-10 Veri setinin kullanımı.

Standartlaştırılmış yapısı ve zengin içeriğiyle CIFAR-10, özellikle derin öğrenme ve makine öğrenimi modellerinin sınıflandırma performansını değerlendirme amacıyla araştırmacılar tarafından yaygın olarak kullanılmaktadır. Bu veri kümesi, sınıflandırma algoritmalarının gerçek dünya problemlerine uyarlanabilirlik performansını değerlendirmek ve farklı tekniklerin verimliliğini nesnel bir biçimde karşılaştırmak için ölçüt işlevi görmektedir.

Tablo 5. CIFAR-10 veri setinin uygulamada kullanacak dijit yapıya dönüştürülmesine ait sözde kod

Input: *data_dir, output_dir*
Output: *patch_train.txt, label_train.txt, patch_test.txt, label_test.txt*
Begin
Function: load_cifar10_batch(file)
    *Open file in **'rb'** mode*
    Load data using pickle and save it to a dictionary
    Extract images (from data key) and labels (from labels key) from the dictionary
    Return images, labels
Function: save_data(images, labels, patch_file, label_file)
    Save images to patch_file
    Save labels to label_file
Function: process_cifar10_data (data_dir, output_dir)
    If output_dir does not exist:
    Create output_dir
    Initialize empty lists: train_images and train_labels
    For each i from 1 to 5:
        Construct file path as data_dir + 'data_batch_i'

```
        Call load_cifar10_batch(file) to load images and labels
        Append images to train_images
        Append labels to train_labels
    Concatenate train_images and train_labels
    Call save_data(train_images, train_labels, 'patch_train.txt', 'label_train.txt')
    Call load_cifar10_batch(data_dir + 'test_batch') to load test_images and test_labels
    Call save_data(test_images, test_labels, 'patch_test.txt', 'label_test.txt') to save test data
Call process_cifar10_data('cifar-10-batches-py', 'cifar10_data')
End
```

Tez çalışmasında önerilen yöntemin performansını değerlendirmek için kullanılan diğer bir veri seti Fisher's Iris veri setidir. Fisher's Iris veri seti, makine öğrenimi ve istatistik alanlarında sıklıkla kullanılan, iyi yapılandırılmış ve basit bir veri setidir. Bu veri seti, her biri dört farklı özellik içeren 150 gözlemden oluşmaktadır ve üç farklı sınıfa ayrılmıştır. Sınıflar, üç farklı iris çiçeği türünü temsil etmektedir. Bunlar; Iris-setosa, Iris-versicolor ve Iris-virginica. Her sınıf, veri setinde eşit sayıda örnek içerir. Fisher's Iris veri seti, denetimli öğrenme tabanlı sınıflandırma çalışmaları için ideal bir ortam sunar. Küçük boyutu ve dengeli sınıf dağılımı sayesinde eğitim ve test işlemleri hızlıca gerçekleştirilebilir. Bunun yanı sıra, özelliklerin doğrusal olarak ayrılabilirliği, algoritmaların performansını analiz etmek için uygun bir zemin oluşturur. CIFAR-10 veri setine uygulanan işlemler python programlama dili ile Fisher's Iris veri setine uygulanarak modelin kullanabileceği binary formata çevrilir. Bu aşamalardan sonra ilgili veri seti önerilen model üzerinde kullanılarak sonuçlar elde edilir.

Memristör Tabanlı Sinir Ağı Donanımı

Donanım tabanlı yapay sinir ağı uygulamaları için memristif sinaptik cihazlar giderek daha fazla benimsenmektedir (J. Chen et al., 2021; Ielmini & Wong, 2018; Qin et al., 2020). Bu cihazlar, biyolojik sinapsların işlevini taklit ederek yapay sinir ağlarında bilgi işleme ve öğrenme sürecini etkinleştirmektedir. Yapay sinir ağları, büyük veri kümelerini kullanarak öğrenme algoritmalarını eğitmekte ve girdi verilerinin temel özelliklerini çıkarımsal olarak öğrenmektedir. Mimari açıdan, yapay sinir ağları genellikle iki ana kategoriye ayrılır. İlk kategori olan ileri beslemeli ağlar, hesaplama sürecini girişten çıkışa doğru, her katmanda ardışık bir şekilde gerçekleştirir. Bu tür ağlar, öğrenme süreci sırasında sinyalleri yalnızca ileri yönde taşıyarak çıktı katmanına ulaşır. Bu ağ yapısı, hesaplama işlemlerini sadeleştirirken doğrusal bilgi akışını korur ve çoğunlukla denetimli öğrenme gerektiren sınıflandırma görevlerinde kullanılır (S. Raj & Ananthi J, 2019). İkinci kategori olan tekrarlayan yapay sinir ağları (RNN), farklı bir yapısal yaklaşım sunar. Bu ağlar, döngüsel bağlantılar aracılığıyla hem ileri hem geri bilgi akışı sağlamakta ve zaman içindeki ardışık ilişkileri dikkate alabilmektedir.

Böylece, veriler arasında bir hafıza etkisi oluşturulmakta ve ağın önceki durum bilgilerini de hesaba katarak daha dinamik bir hesaplama kapasitesine sahip olması sağlanmaktadır. Bu çalışmada ikinci kategoride belirtilen yapay sinir ağı modeli kullanılmıştır. İleri beslemeli bir sinir ağında nöronlar, katmanlar halinde düzenlenmiştir; her katmandaki nöronların çıktısı bir sonraki katmana ağırlıklı bir giriş olarak aktarılır ve bu süreç, katmanlar arası sinirsel iletimin temelini oluşturur.

Tablo 6, memristör tabanlı donanım uygulamalarına yönelik yapay sinir ağları üzerine gerçekleştirilen dört farklı çalışmanın karşılaştırmalı analizini sunmaktadır. Bu çalışmalar, biriktirme yöntemleri, memristör yapıları, aktif katman kalınlıkları, R_{OFF}/R_{ON} oranları, sinaptik özellikler, iletkenlik durumları ve sinaptik voltaj değerlerindeki farklılıklar açısından detaylı bir şekilde incelenmiştir. Analiz, memristörlerin tasarım ve performans parametrelerinin sinir ağı uygulamalarındaki etkilerini anlamaya yönelik kapsamlı bir bakış açısı sağlamaktadır. Referanslarda belirtilen çalışmalarda ve bu tez çalışmasında, biriktirme yöntemleri olarak Darbeli Lazer Biriktirme (PLD), Plazma Destekli Atomik Katman Biriktirme (PEALD), Reaktif Magnetron Püskürtme ve RF-Magnetron Püskürtme kullanılmıştır. Çalışmalar, 10 nm ile 65 nm arasında değişen aktif katman kalınlıklarına odaklanmış ve farklı R_{OFF}/R_{ON} oranları rapor edilmiştir. Sinaptik özellikler bu çalışma ve (L. Gao et al., 2015; Illarionov et al., 2020; Miyake et al., 2022) referanslarında ayrıntılı olarak açıklanmıştır. Çeşitli çalışmalar, iletkenlik durumlarının 102 ile 210 arasında değişkenlik gösterdiğini ve sinaptik voltajların 3 V ile 10 V aralığında dalgalandığını bildirmiştir.

Tablo 6. Farklı memristör tabanlı donanım uygulaması yapay sinir ağı çalışmaları.

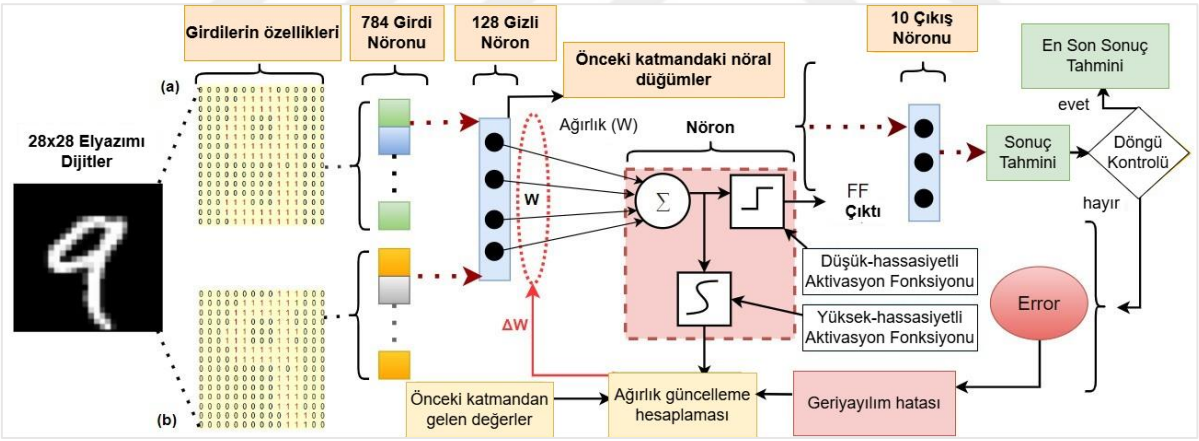
Çalışma	(Miyake et al., 2022)	(Illarionov et al., 2020)	(L. Gao et al., 2015)	Bu çalışma
Biriktirme yöntemi	PLD (Darbeli Lazer biriktirme)	PEALD (Plazma Destekli Atomik Katman Biriktirme)	Reaktif Magnetron Püskürtme	RF-Magnetron Sputtering
Yapı	Pt/TiO2−x/Pt	Al/TiO2/Al	Ta/TaOx/TiO2/Ti	Al/TiO2/Al
Aktif katman kalınlığı	65 nm	13 nm	10 nm+30 nm	10 nm
R_{OFF}/R_{ON} ratio(RRAM)	20	100	Rapor edilmedi	> 2
Sinaptik özellikler	+	Rapor edilmedi	+	+
İletkenlik durumları	Not reported	Rapor edilmedi	102	210
Sinaptik gerilim	7 V, 8 V, 9 V, 10 V	Rapor edilmedi	3 V	5 V

Memristör Sinaptik Tabanlı Cihaz Kullanarak Nöral Ağ Uygulaması

Memristif sinaptik cihazlar, gelişmiş öğrenme algoritmaları ve yüksek kaliteli veri kümelerinin yardımıyla rakam tanıma ve görüntü sınıflandırma gibi görevlerde performansı önemli ölçüde artırmaktadır. Bu cihazlar, yapay zekâ ve makine öğrenimi uygulamaları için

ideal bir yapı sunarak, düşük güç tüketimi ve yüksek işlem hızı avantajları ile öne çıkmaktadır. Çalışmanın odağı, çeşitli optimizasyon algoritmaları kullanılarak hız, enerji verimliliği ve doğruluğun optimize edilmesidir. Bu çerçevede, memristör tabanlı nano-sinaptik cihazların sahip olduğu özelliklerin ve mevcut öğrenme performanslarındaki sınırlılıkların ele alındığı bir nöro-esinli donanım çözümü sunulmaktadır.

Bu çalışmada, memristör tabanlı cihazın memristif özelliklerini kullanarak ileri besleme (FF) ve geri yayılım (BP) yöntemleriyle performans ve verimliliği artırma hedeflenmiştir. Performans karşılaştırmaları ve referans sağlamak amacıyla iki katmanlı bir perceptron (MLP) sinir ağı tercih edilerek tasarlanmıştır. Şekil 21(a)'da gösterildiği gibi, bu sinir ağı, giriş, gizli ve çıkış katmanlarından oluşmakta ve her katmandaki nöronlar, bir sonraki katmandaki tüm nöronlara tam bağlantılarla bağlanmaktadır. Bu bağlantı yapısı, ağın karmaşık veri modellerini öğrenme kapasitesini artırırken, ağırlıklı sinapslarla temsil edilen esnek bağlantılarla yüksek hesaplama gücü sunmaktadır. Giriş ve gizli katmanlar arasındaki bağlantı ağırlıkları matris W_{IH} ile, gizli ve çıkış katmanları arasındaki ağırlıklar ise matris W_{HO} ile gösterilmektedir. Ağın eğitimi için giriş verisi olarak 28x28 piksel boyutuna yeniden ölçeklendirilmiş MNIST el yazısı rakamları ve CIFAR-10 veri setleri kullanılmıştır.



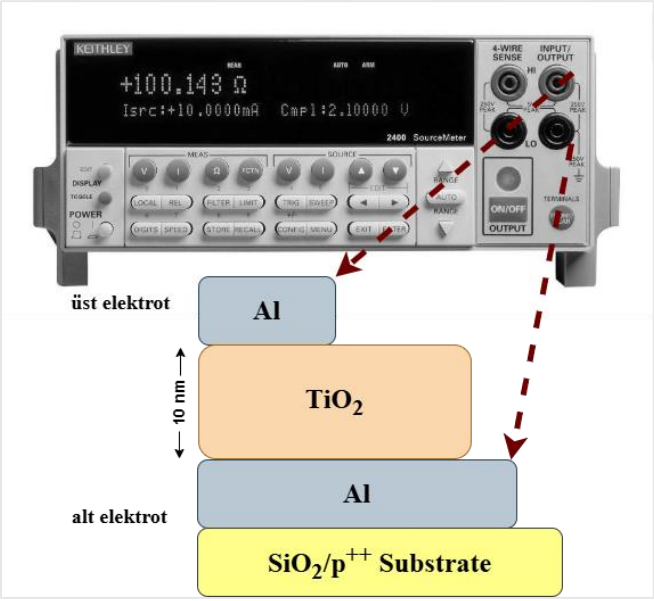
Şekil 21. MLP nöral ağı. Kullanılan BPNN'nin mimarisi ve nöronları. (a) Bilgisayar tarafından üretilen bir rakamı temsil eden ikili matris (28x28) ve (b) el yazısı rakam.

Ağ topolojisinin varsayılan yapısı 784 nöronlu bir giriş katmanı, 128 nöronlu bir gizli katman ve 10 sınıfa karşılık gelen 10 nöronlu bir çıkış katmanı içermektedir. Bu yapı, her bir görüntüyü işleyerek uygun rakam sınıfını belirleme yeteneğini sağlar. Bu parametrelerde yapılacak değişiklikler, ağın performansını optimize etmek için yeni ayarların yapılmasını gerektirebilir. Şekil 21(b) ise bir nöron düğümünü gösterir; burada nöron, gelen sinapslardan ağırlıklı bir toplamı hesaplamakta ve 1 bitlik düşük hassasiyetli bir aktivasyon fonksiyonu ile çevrimdışı sınıflandırma gerçekleştirmektedir. Ancak, eğitim sürecinde geriye yayılım

yapılırken, küçük hata düzeltmeleri için daha yüksek hassasiyetli ağırlık güncellemeleri gerekmekte olup çalışmada bu durum göz önünde bulundurularak işlemler gerçekleştirilmiştir.

Bu tez çalışmasında tasarlanan memristör tabanlı sinaptik cihazın performansını artırmak amacıyla çeşitli optimizasyon yöntemleri uygulanmıştır. Bu cihazda, öğrenme süreci sırasında oluşan sinaptik ağırlık değişimlerinin ölçülmesi ve izlenmesi için özel bir prob istasyonu kullanılmıştır. Ayrıca, memristör cihazının sinaptik davranışlarının detaylı analizleri, bu amaca yönelik özel olarak geliştirilmiş bir yazılım ile yürütülerek sonuçlandırılmıştır.

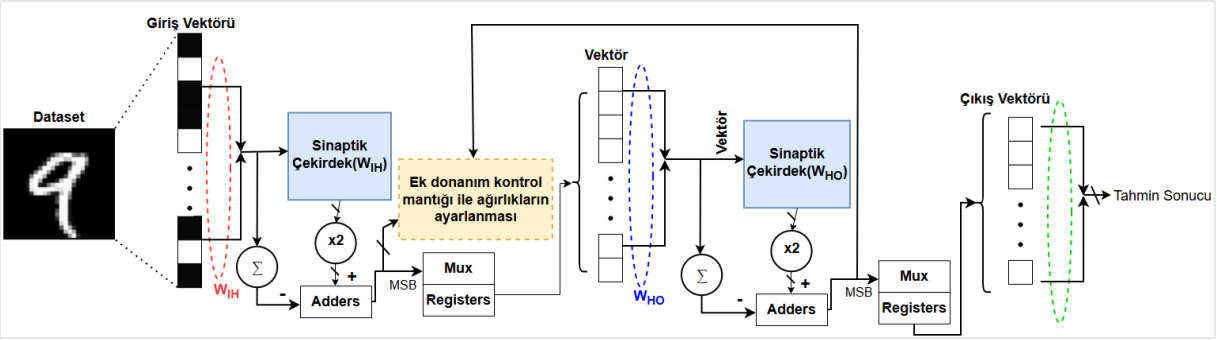
Memristif eNVM sinaptik çekirdeklerindeki enerji tüketiminin ana kaynağı, geleneksel belleklerde sıklıkla görülen dinamik güç tüketiminden farklı olarak, daha çok statik güç tüketimine dayanır. Memristif sinapslardan geçen elektrik akımı, enerji tüketimine önemli bir katkı sağlamakta olup toplam enerji tüketimi hem sinaptik çekirdekler hem de çevresel devrelerin tükettiği enerjinin bileşimi olarak hesaplanmaktadır. Çevresel devrelerin enerji tüketimi hesaplamaları, 32 nm düğüm teknolojisi ile öngörücü teknoloji modeli (PTM) aracılığıyla gerçekleştirilmiştir.



Şekil 22. SMU/Pulse Source. Memristör sinaptik cihazın yerleşimi ve deneysel kurulum

Sinir ağı uygulamasının bilgisayar üzerinde enerji tüketimi analizi için, i7-10750H işlemci (2.90 GHz) ve 8 GB RAM'e sahip bir sistem kullanılmıştır. Bu analiz sırasında enerji tüketimi ölçümlerinde (García-Martín et al., 2019) kaynakta sunulan yönergeler temel alınmış ve sinir ağı uygulamasının bilgisayardaki enerji kullanımını tahmin etmek amacıyla DeLight adlı analiz aracı (Şekil 22'de gösterilen) kullanılmıştır. Bu yöntemler, sinir ağlarının donanım

tabanlı uygulamaları için daha düşük enerji gereksinimleriyle yüksek verim elde etme potansiyelini değerlendirmede kritik bir rol oynamaktadır.



Şekil 23. İki katmanlı MLP ağının donanımda uygulanması için devre blok diyagramı.

Şekil 23, iki katmanlı MLP sinir ağının donanım tabanlı uygulaması için devre blok diyagramını göstermektedir. Bu uygulamada, ağırlıklı toplam hesaplamaları sinaptik çekirdekler aracılığıyla gerçekleştirilir. Ancak, standart bir sinaptik dizide kullanılan ağırlıklar sadece W_H = *0~1* aralığında negatif olmayan değerler alabilirken, sinir ağı algoritmasında hem pozitif hem de negatif ağırlık değerleri mevcuttur, yani W_A = *-1~1*. Bu durum donanım tasarımında bazı zorluklar ortaya çıkarmakta ve negatif ağırlıkları temsil etmek için özel çözümler gerektirmektedir. Bu uygulamada, ağırlıkları W_A=*-1~1* aralığından W_H = *0~1* aralığına dönüştürmek için algoritmik bir yapı kullanılmıştır. Algoritmadaki ağırlıklı toplam hesaplaması aşağıdaki gibi ifade edilir:

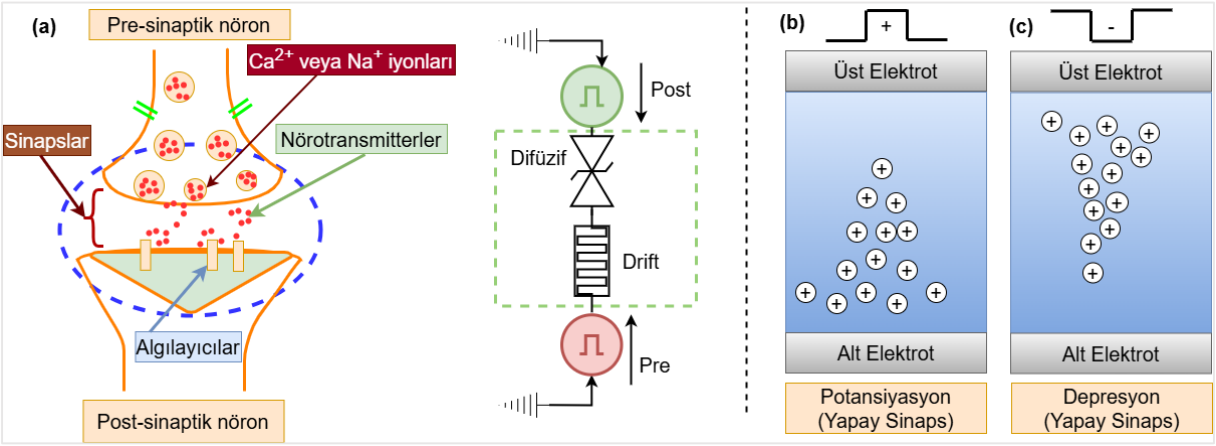
$$W_A V = 2(W_H - 0.5J)V = 2W_H V - JV) \tag{6}$$

Bu ifadede V giriş vektörünü, J ise W_A ve W_H ile aynı boyutta ve tüm elemanları bire eşit olan bir matrisi temsil etmektedir. Denklem (6)'da $W_H V$, sinaptik çekirdekten elde edilen ağırlıklı toplamın sonucunu temsil eder. Bu nedenle W_A, *(-1~1)* aralığından W_H *(-1~0)* aralığına dönüştürülür. Özetle, Şekil 21 bir sinir ağının donanımsal mimarisini göstermektedir. MNIST, CIFAR-10 ve Fisher's Iris veri kümelerinin bir kısmı girdi vektörü olarak kullanılır ve çıktı vektörünü tahmin etmek için sinaptik çekirdekler *(W_{IH} ve W_{HO})* aracılığıyla işlenir. Ara katmanda, ağırlıklar donanım kontrol mantığı tarafından ayarlanır ve MSB tarafından işlenir. Sonuç olarak, çıktı katmanı tahmini sağlar. Şekil 23 sinir ağlarının donanım uygulamasını göstermektedir.

Cihaz Özellikleri ve Uygulama Deneyleri

Malzeme ve cihaz özellikleri

Memristörlere dayalı nöromorfik hesaplama sistemleri, yazılım tabanlı sinir ağlarından daha enerji verimlidir. Doğal analog dirençler gibi davranan memristörler, hesaplamaları bellek içinde gerçekleştirerek von Neumann mimarisine bir alternatif sunar. Büyük ölçekli memristör tasarımlarını tahmin ve optimize etmek için NeuroSim ile simülasyonlara ihtiyaç vardır. İkili oksitler, basit üretimleri ve CMOS teknolojisi ile uyumlulukları nedeniyle elektronik memristörler için çok önemlidir (Ye et al., 2022). Bu memristörlerin üretimi kolay, düşük maliyetli ve mevcut yarı iletken teknolojileriyle iyi entegre olabilirler. Hf, Zn, V, Ni ve Ti (Gale, 2014) gibi oksitler, kademeli direnç anahtarlamaları için sinaptik nano cihazlarda kullanılmaktadır. Bu sinaps benzeri davranış, memristörleri nöromorfik hesaplama ve sinir ağı uygulamaları için ideal hale getirmektedir. Ayrıca sinaptik işlevleri taklit eden nano-elektronik cihazların geliştirilmesine önemli bir ilgi vardır (Kuzum, 2018). Memristör, CMOS üretimindeki önemi nedeniyle nöromorfik bilgi işlem için umut vericidir. Memristör tabanlı sinaptik cihazlar, bellek içi hesaplama için çapraz çubuk mimarilerine zahmetsizce dahil edilebilir. Şekil 24(a) biyolojik sinaps muadillerini ve memristörün anahtarlama davranışını detaylandırmaktadır. Yapay sinaps oluşturma, bipolar elektrotlar arasında yer alan nanoparçacıkların elektrokimyasal reaksiyonlarına dayanır. Gerilim uygulaması iyon ve atom göçüne neden olarak iki kutuplu anahtarlama sağlar (Ilyas et al., 2020).



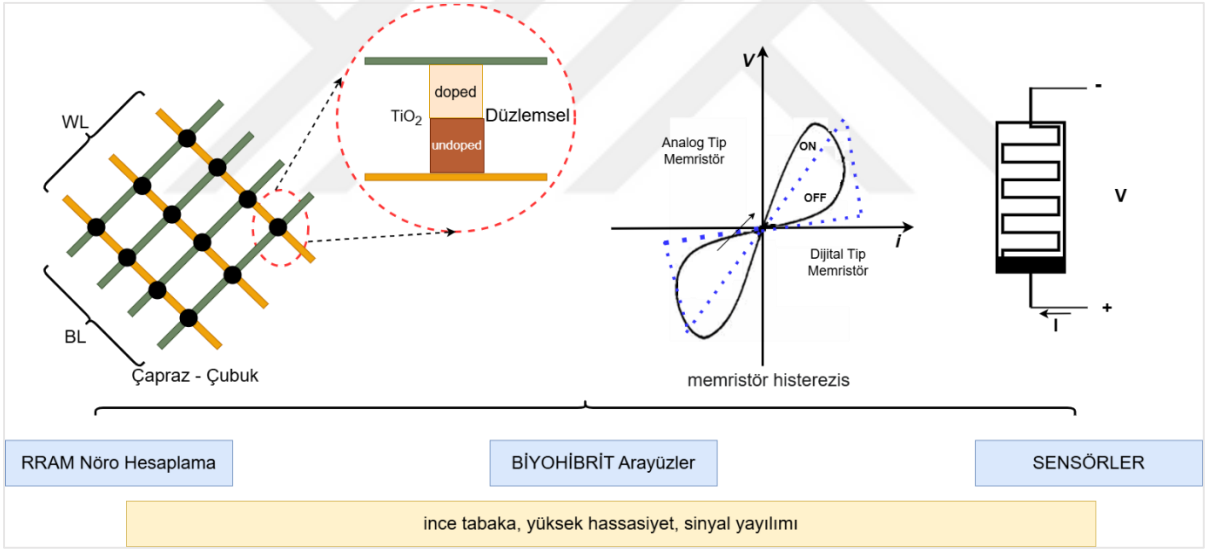
Şekil 24. Biyolojik ve yapay sinapsların şematik gösterimi. Nörotransmitterler ve reseptörlerle biyolojik bir sinaps gibi, PRE ve POST sivri uçları burada da kullanılır

Şekil 24(c), voltajın nanoparçacıkların büyümesine, akımın artmasına ve memristörün düşük dirençli bir duruma geçmesine neden olduğunu göstermektedir. Gerilim geri çekildiğinde, Şekil 24(b)'de gösterildiği gibi, nanoparçacıklar küçülür, akımı azaltır ve memristörü yüksek dirençli

bir duruma geçirir. Memristör tabanlı cihazların direnci dinamik olarak değiştirebilir ve biyolojik sinapsların işlevselliğini taklit edebilir. Bu cihazlar nöromorfik mühendislik ve yapay sinir ağları gibi ileri teknolojik uygulamalar için büyük bir potansiyele sahiptir.

Memristör tabanlı sinaptik cihazın özellikleri

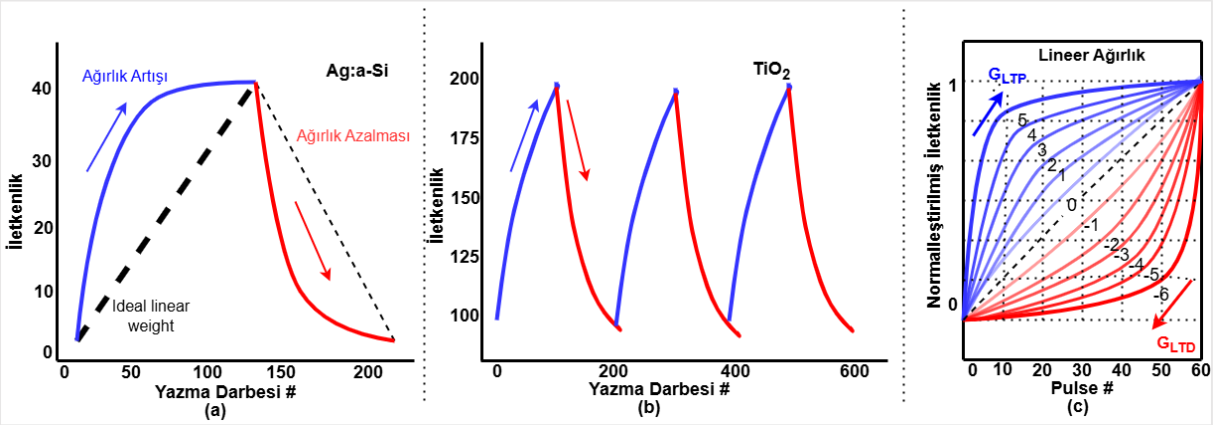
Memristörlerin gelişimi RRAM, biyohibrit sistemler ve sensörler gibi yüksek teknoloji uygulamalarıyla bağlantılıdır. Bu uygulamalar memristörlerin potansiyeli Şekil 25'te gösterilmiştir. TiO_x, RRAM uygulamaları için çalışılan en eski malzemeler arasındadır. Bununla birlikte, memristörler genellikle *AÇIK* veya *KAPALI* durumlar arasında değişen dirençli anahtarlama bellek cihazları olarak kullanılır (Wong et al., 2012). Memristör tabanlı RRAM cihazları, iki durum arasında geçiş yaparak veri depolar ve güç kapalı olsa bile gerektiğinde erişilebilir kalır (Q. Wan et al., 2019). Bazı nano ölçekli metal oksit memristörler, dirençte ince ayarlara izin vererek hassas veri işleme ve depolamaya olanak tanır. Bu da onları gelişmiş nöromorfik hesaplama ve yenilikçi bellek uygulamaları için ideal hale getirmektedir.



Şekil 25. Memristif Cihazların Teknolojik Gelişmeleri ve Uygulamaları

Teorik olarak, sinaptik ağırlık değişiklikleri yazma darbelerinin sayısıyla doğrusal olarak ilişkili olmalıdır, ancak gerçek dünyadaki cihazlar bu davranıştan sapmaktadır. İletkenlik *LTP* ve *LTD*'nin ilk aşamalarında hızla değişir ve kademeli olarak doygunluğa ulaşır. Şekil 26 ((a), (b) ve (c)) nöromorfik sistemlerin doğruluğunu ve verimliliğini etkileyen bu sapmaları göstermektedir. Bu da detaylı karakterizasyon ve optimizasyon gerektirmektedir. Doğrusal olmayan ağırlık güncellemelerini modellemek için bir cihaz modeli tasarlanmıştır. MLP + NeuroSim, memristör tabanlı sinaptik cihaz gibi analog eNVM sinapslarına sahip nöromorfik

sistemlerin güç kullanımını, eğitim gecikmesini ve uzamsal gereksinimlerini modellemektedir (Luo et al., 2019).



Şekil 26. Ag:a-Si için doğrusal olmayan ağırlık güncellemesi, (b) TiO_2 için doğrusal olmayan ağırlık güncellemesi, (c) Analog eNVM cihaz davranış modelinin -6'dan 6'ya kadar etiketlenmiş doğrusal olmayan ağırlık güncellemeleriyle şematik gösterimi

İletkenlik değişimi yazma darbeleri (P) ile ilişkilidir ve aşağıdaki denklemlerle ifade edilir:

$$G_{LTP} = B\left(1 - e^{\left(-\frac{P}{A}\right)}\right) + G_{min} \tag{7}$$

$$G_{LTP} = -B\left(1 - e^{\left(-\frac{P-P_{max}}{A}\right)}\right) + G_{max} \tag{8}$$

$$B = (G_{max} - G_{min})/(1 - e^{-P_{max}/A}) \tag{9}$$

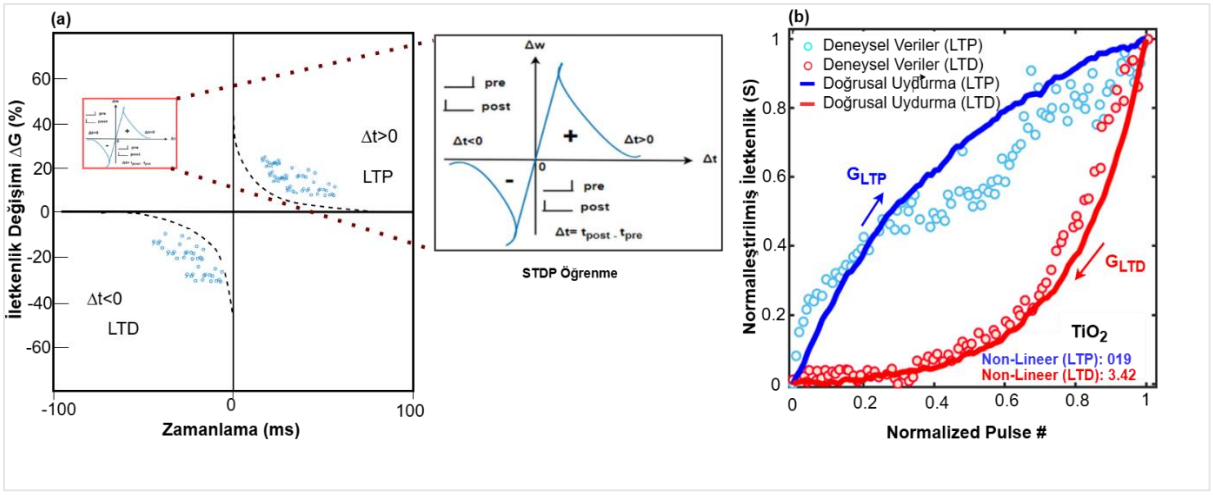
G_{LTP} ve G_{LTD} sırasıyla LTP ve LTD süreçleri için iletkenlik değerlerini temsil eder. G_{max}, G_{min} ve P_{max} doğrudan deneysel sonuçlardan elde edilen değerlerdir ve sırasıyla en yüksek iletkenliği, en düşük iletkenliği ve cihazın en yüksek ve en düşük iletkenlik durumları arasında geçiş yapması için gereken maksimum darbe sayısını ifade eder. *A* parametresi ağırlık güncelleme davranışının doğrusal olmayan doğasını kontrol eder ve pozitif (mavi) veya negatif (kırmızı) olabilir. Şekil 24(a) ve Şekil 24(b)'de LTP ve LTD için A değerlerinin büyüklükleri aynıdır, ancak işaretleri farklıdır. B, A'nın bir fonksiyonu olarak tanımlanır ve G_{max}, G_{min} ve P_{max} kapsamındaki fonksiyonları uydurmak için kullanılır. Denklem ((7), (8) ve (9)) kullanılarak, Şekil 24(c)'de gösterildiği gibi A değeri ayarlanarak çeşitli doğrusal olmayan ağırlık artırma (mavi) ve ağırlık azaltma (kırmızı) davranışları elde edilebilir. Her bir doğrusal olmayan eğri +6 ile -6 arasında değişen doğrusal olmama değerleriyle etiketlenmiştir (P.-Y. Chen et al., 2018). Denklem (7) ve (8) incelendiğinde, A'nın işareti dışında eşdeğer oldukları kanıtlanabilir. Bu nedenle hem doğrusal olmayan LTP hem de LTD ağırlık güncellemeleri için yalnızca Denklem (7) kullanılacaktır. Şekil 26(c)'den ((a) ve (b)) farklı olarak, daha basit

formülasyon için eğrilerin sıfır numaralı darbeden başlamasını sağlamak amacıyla tüm LTD eğrileri yansıtılmış ve yatay olarak kaydırılmıştır.

Biyolojik sinir ağlarında, sinaptik ağırlık STDP ile modüle edilir ve hücre iletkenliği ile ölçülür (Prezioso et al., 2016). Bu, yapay sinir ağlarındaki öğrenmeyi taklit eder. Memristör cihazın sinaptik ağırlık değişimi, tepe zamanlaması darbelerinden kaynaklanan iletkenlik değişiklikleri ile belirlenir, bu da ön ve ardışık sinaptik tepe zamanlamasına bağlı olarak güçlendirme veya zayıflamaya yol açar. Memristör tabanlı memristif cihazlar, biyolojik sinapsların işlevselliğini taklit edebilir. Sinaptik ağırlık değişiminin miktarı ve yönü, tepe zamanlamaları arasındaki göreceli iletkenlik değişimi ile bağlantılı olup STDP ilkeleri ve sinir ağı öğrenme mekanizmalarını yansıtır. Bu ifadenin matematiksel olarak nasıl tanımlandığını belirtmek için Denklem (10) şu şekildedir:

$$\Delta G = (G_{after} - G_{before})/G_{before} \tag{10}$$

Bu denkleme göre, ΔG sinaptik ağırlık değişimini temsil ederken, G_{after} ve G_{before} sırasıyla ön-son sivri uçların aktivasyonunu takip eden ve öncesindeki iletkenlik değerlerini temsil eder (S. Yu et al., 2011). Bu oran sinaptik modülasyon etkinliğini ve yönünü belirler. Şekil 27(a), memristör tabanlı sinaptik cihazın spike-time bağlı plastisite (STDP) özelliklerini sunarak sinaptik öncesi ve sonrası spike uçlar arasındaki zamanlamanın sinaptik iletkenlikteki değişiklikleri nasıl etkilediğini ifade etmektedir. STDP mekanizması, bağlantıları güçlendiren veya zayıflatan sinaptik gücün modüle edilmesinde önemli bir rol oynamaktadır. Memristörler biyolojik sinapsların davranışını taklit etmede ve nöromorfik sistemlerde uygulanabilirliklerinde umut vaat etmektedir. Şekil 27(b) cihazın çeşitli zamanlama aralıklarındaki performansını göstermekte ve sinaptik ağırlık modifikasyonunun dinamiklerini ortaya koymaktadır.



Şekil 27. Memristör Tabanlı Sinaptik Cihazın Spike-Zamanlamasına Bağlı Plastisite (*STDP*) Özellikleri. (b) Memristör tabanlı ağırlık ayarlama verileri için normalleştirilmiş darbe (pulse) sayısı ve normalleştirilmiş iletkenlik arasındaki ilişkiyi gösteren grafik

MLP + NeuroSim, çevrimiçi öğrenme süreçlerini karşılaştırmak için yaygın olarak kullanılmaktadır. Memristör tabanlı analog *eNVM* sinapsları ile nöromorfik donanımın güç tüketimi, eğitim gecikmesi ve alan kullanımı gibi metriklerini simüle eder. Normalleştirme parametreleri, deneysel ağırlık verilerinin yeniden düzenlenmesi ve normalleştirilmesiyle belirlenmiştir. *LTP* ve *LTD* verileri NeuroSim doğrusal olmayan uydurma yöntemi kullanılarak yansıtılmış ve uydurulmuş bu işlemin sonucunda sırasıyla 0,19 ve 3,42 doğrusal olmayan değerlere ulaşılmıştır. Bu değerler simülatörde kullanılmış ve sonuçlar Şekil 27(b)'de gösterilmiştir. Bu simülasyonlar memristör tabanlı nöromorfik donanımın performansını optimize etmeye yardımcı olmaktadır.

Yapay sinaps olarak memristör ve uygulamanın yazılım-donanım entegrasyonu

Gelişen teknolojiyle birlikte yapay zekâ ve sinir ağları, enerji verimliliği ve hesaplama performansı açısından daha optimize çözümlere duyulan ihtiyacı artırmıştır. Geleneksel hesaplama yaklaşımlarının ötesine geçerek, memristör tabanlı yapay sinapslar, enerji verimli ve biyolojik ilhamlı donanım mimarileri için umut vaat eden bileşenler olarak ortaya çıkmıştır. Bu bölümde, memristörlerin yapay sinir ağı uygulamalarındaki rolü ve yazılım-donanım entegrasyonu ile bu sistemlerin etkinliği ele alınmaktadır.

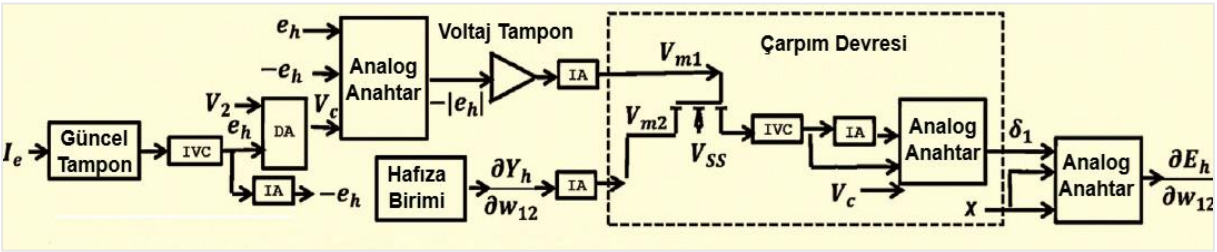
Memristör, elektriksel direnci hafızasına alabilen bir bileşendir ve doğrudan biyolojik sinapsları taklit etmek için kullanılabilir. Bu özellik hem verilerin depolanması hem de işlenmesi için aynı fiziksel yapıyı kullanması nedeniyle geleneksel transistörlere kıyasla çok daha verimli çözümler sunar.

Bu tez çalışmasında memristör özelliği gösteren TiO_2 tabanlı malzemeler kullanılmıştır. TiO2 tabanlı sinaptik bileşen, laboratuvar ortamında tasarlanarak bilgisayar ortamında doğrusal olmayan değerleri elde edilmiştir. Elde edilen bu parametreler ile memristör tabanlı bir yapay sinir ağının çalışması bilgisayar ortamında simüle edilmiştir. Ayrıca yapay sinapsların etkili bir şekilde kullanılabilmesi için, donanım mimarisi ile yazılım algoritmalarının uyumlu bir şekilde entegre edilmesi gerekmektedir. Bu entegrasyon, ağırlıkların donanım seviyesinde doğrudan temsil edilmesi ve işlem süreçlerinin donanım tabanlı olarak optimize edilmesini sağlamaktadır. Tasarım aşamasından sonra uygulamanın yazılım-donanım entegrasyon aşaması gerçekleştirilmiştir.

Çalışmamızda ilk önce ağırlıkların donanıma haritalanması yapılmaktadır. Memristör dizileri, sinir ağındaki ağırlıkların fiziksel olarak depolandığı yapılardır. Yazılım tarafından öğretilen ağırlıklar, bu dizilere aktarılarak hesaplama işlemleri doğrudan donanım üzerinde gerçekleştirilir.

Yazılım-donanım entegrasyonunun ikinci aşamasında donanım tabanlı hesaplama gerçekleştirilir. Gerçekleştirilen çalışmada yapay sinir ağlarında hesaplama işlemleri, akım ve gerilim sinyalleri üzerinden gerçekleştirilir. Crossbar mimarisi, çarpma ve toplama işlemlerinin paralel olarak yapılmasını sağlar. Bu, yazılım tabanlı işlemlere kıyasla hem zaman hem de enerji verimliliği sunar.

Uygulamanın üçüncü aşmasında hata düzeltme ve geri yayılım işlemi için yazılım-donanım entegrasyonu ayarlanır. Bu işlem yapılarak eğitim sürecinde, memristörlerin doğal gürültü ve hassasiyet sorunlarının etkisi azaltılmaktadır. Çalışmada kullanılan yazılım algoritmalarının, hata sinyallerini analiz ederek ağırlık güncellemelerinin donanıma yansıtılması sağlanır. Son olarak uygulamanın donanım düzeyindeki enerji verimliliği yazılım-donanım entegrasyonu ile analiz edilir.



Şekil 28. Geriye Yayılım Devresi.

Şekil 28’de, sinir ağlarının sinaptik entegrasyon sürecinde kritik bir bileşen olan Geri Yayılım Devresi'nin (BPC) karmaşık işleyişini detaylandırmaktadır. Hata terimlerinin gizli katmanlara iletilmesinden, CROSSBAR (CB) yapısındaki son ağırlık güncellemelerine kadar her aşama titizlikle incelenmiştir. Bu çalışmada özellikle, NMOS transistörü M_{60} etrafında yapılandırılan Çarpma Devresinin doğrusal çarpma işlemlerini gerçekleştirmedeki rolü ve farklı giriş konfigürasyonlarını nasıl ele aldığı kapsamlı bir şekilde ele alınmıştır. Ayrıca, bu devrede Analog Anahtarlar (AS) ve Ters Çeviren Yükselteçlerin (IA) kullanımı, devrenin karmaşıklığına ek bir boyut katmaktadır. Son olarak, devre çıkışlarının ağırlık güncelleme mekanizmasındaki kritik rolü vurgulanmış; genlik ve işaretin ağırlık ayarlamalarını nasıl şekillendirdiği üzerinde durulmuştur.

Bu çalışma, TiO_2 tabanlı memristörlerin yapay sinir ağlarında kullanımına yönelik özgün bir yaklaşım sunmaktadır. Özellikle, laboratuvar ortamında tasarlanan sinaptik bileşenlerin bilgisayar ortamında doğrusal olmayan parametrelerinin elde edilmesi ve bu verilerin memristör tabanlı bir yapay sinir ağının simülasyonlarında kullanılması, literatürde sınırlı sayıda ele alınmış bir konudur. Bununla birlikte, yazılım ve donanım entegrasyonunun her üç aşamasının detaylı bir şekilde ele alınması ve memristörlerin doğal gürültü ve hassasiyet sorunlarına karşı alınan önlemler, çalışmanın yenilikçi yönlerini oluşturmaktadır. Ayrıca TiO_2 tabanlı sinaptik cihazın kullanıldığı yapay sinir ağlarında optimizasyon yöntemlerinin doğruluk, enerji ve hız bakımından karşılaştırıldığı, bununla birlikte önerilen modelin çok sayıda veri seti ile kararlılığının test edildiği çalışmaya rastlanmamıştır. Bunlar çalışmanın özgünlüğünü pekiştirmektedir.

Uygulamada kullanılan optimizasyon algoritmalarının sınıflandırılması

Optimizasyon algoritmaları doğruluk, hız ve genelleme performansı gibi hedeflere göre sınıflandırılır (Bian & Priyadarshi, 2024). Doğruluk, modelin doğru tahminler yapabilme yeteneğini; hız, işlem süresini; genelleme performansı ise modelin yeni verilere uyarlanabilirliğini göstermektedir. Genelleştirme performansı optimizasyon algoritmalarında büyük ve etkin bir öneme sahip olup parametrelerin kolayca güncellenebilmesi bu algoritmaları daha pratik hale getirir. Ayrıca, büyük veri kümelerinde enerji tüketimi performansı da önemli bir faktördür. Farklı bileşenler farklı optimizasyon teknikleri ve stratejileri gerektirmektedir. Bu çalışmada, Stocastic Gradient Descent (SGD) ve varyasyonları makine öğrenimi ve derin öğrenme uygulamalarında kullanılmış olup, bu algoritmalar ilerleyen bölümlerde kısaca açıklanmıştır. Bu çalışma, memristör sinaptik tabanlı bir cihaz kullanarak nöromorfik hesaplama alanında sekiz optimizasyon yöntemini karşılaştırmaktadır.

Kullanılan optimizasyon yöntemleri ve eğitim aşaması

Gradyan Descent ve türleri

Yapay sinir ağlarını eğitmek, kayıp fonksiyonunu en aza indiren ağ parametrelerini belirlemeyi amaçlayan bir optimizasyon sürecidir. Bu öğrenme modelleri büyük veri kümeleri ve ayarlanması gereken çok sayıda model parametresi gerektirir. Büyük veri kümelerini işlemek için en uygun yöntemi bulmak önemli bir engel oluşturmaktadır. Gradyan inişi, model parametrelerini gradyanın ters yönünde düzenli olarak güncelleyerek kayıp fonksiyonlarını azaltmak için çok önemli bir yöntemdir (Bian & Priyadarshi, 2024). Gradyan inişinin çeşitli versiyonları, optimizasyon sürecinde karşılaşılan zorlukların üstesinden gelmek için farklı algoritmik stratejiler sunar.

Makine öğreniminde kullanılan birçok model bulunmaktadır. Bunlardan biri olan Stokastik Gradyan İnişi (SGD), parametreleri optimize ederek bir model oluşturur ve bu model aracılığıyla geleceğe yönelik tahminler yapar (Ruder, 2016). Büyük veri setlerinde hızlı ve etkili bir optimizasyon sağlar. Bu nedenle SGD ve varyantları olan algoritmalar çalışmamız için uygun optimizasyon yöntemleridir. Aşağıda SGD ve varyasyonlarının bir özeti verilmiştir. Bu özetlerle beraber tez çalışmasında kullanımı ile ilgili hesaplama ve kod yapıları gösterilmiştir.

Stochastic Gradient Descent (SGD)

SGD (Robbins & Monro, 1951), makine öğrenimi ve derin öğrenme optimizasyon problemlerini çözmek için sıklıkla önerilmektedir. SGD, basit, anlaşılır ve yapay zekâ modellerinde etkilidir. SGD için matematiksel Denklem 11'de gösterilmiştir.

$$\theta_{t+1} = \theta_t - \alpha \nabla F_i(\theta_t) \tag{11}$$

Denklem 11'de; θ_t, t adımındaki ağırlık vektörüdür. α, her adımda ne kadar hareket edeceğimizi belirleyen öğrenme oranıdır. $F_i(\theta_t)$ Gradyan vektörüdür, rastgele seçilen i alt kümesi için kayıp fonksiyonunun gradyanıdır. Bu denklemde, her iterasyonda rastgele bir veri alt kümesi seçilir ve bu alt kümedeki örnekler için gradyan hesaplanır. Bu gradyan, ağırlık vektörünü güncellemek için ağırlık vektörünün mevcut değerinden çıkarılır**.** Tablo 7'te SGD algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 7. Tez çalışmasında kullanılan SGD yönteminin eğitim aşamasındaki sözde kodu

***Input**: numTrain, epochs, learningRate, initialWeights*
***Output**: bestWeights, bestLoss*
Begin

```
weights ← initialWeights
bestWeights ← weights
bestLoss ← ∞
for epoch ← 1 to epochs do
    for batch ← 1 to numTrain do
        // İleri yayılım
        hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
        outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
        // Hata hesaplama
        error ← Output - outputLayerOutput
        // Hata geri yayılımı
        outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
        hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) *
                              ActivationDerivative(hiddenLayerOutput)
        // Ağırlık güncelleme
        deltaWeightOutput ← -learningRate * outputLayerGradient * hiddenLayerOutput^T
        deltaWeightHidden ← -learningRate * hiddenLayerGradient * Input^T
        weights.outputLayer ← weights.outputLayer + deltaWeightOutput
        weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
        // Eğitim kaybının hesaplanması
        currentLoss ← LossFunction(error)
        if currentLoss < bestLoss then
            bestWeights ← weights
            bestLoss ← currentLoss
        End if
    End for
End for
return bestWeights, bestLoss
End
```

Tez çalışmasında SGD yöntemi memristör tabanlı bir uygulama için kullanılmıştır. SGD optimizasyon yöntemi uygulamada kullanılan veri setlerine ait her eğitim örneğine göre ağırlıkları güncelleyerek klasik Gradient Descent algoritmasına kıyasla daha hızlı yakınsama sağlar. SGD algoritması, tüm veri kümesi yerine her adımda yalnızca bir örnek veya küçük bir batch kullanarak ağırlıkları günceller. Bu özellik, SGD'nin büyük veri kümeleriyle hızlı bir şekilde çalışmasına olanak tanır. SGD'nin temel yaklaşımı, model parametrelerini güncellemek için her adımda küçük bir veri grubuna dayalı gradyan bilgisini kullanmaktır. *Train.cpp* dosyasındaki SGD algoritması, modelin ağırlıklarını *initialWeights* ile başlatır ve en iyi ağırlıkları *bestWeights* olarak saklar. En iyi kayıp değeri (*bestLoss*), başlangıçta ∞ olarak ayarlanır, böylece eğitim süreci boyunca ilk güncellenen değer en düşük olarak kabul edilir.

Eğitim döngüsü, belirli bir epoch sayısı boyunca devam eder. Her epoch içinde model, tüm batch'ler üzerinde ileri ve geri yayılım yaparak güncellenir. İleri yayılım sırasında, modelin girdisi gizli katman ağırlıklarıyla çarpılarak *hiddenLayerOutput* elde edilir. Bu ara katman çıktısı, çıkış katmanına gönderilerek *outputLayerOutput* hesaplanır, bu da modelin tahmin

çıktısını temsil eder. Bu tahmin, modelin performansını ölçmek için kullanılır ve hata değeri (*error*), tahmin edilen çıkış ile gerçek çıkış arasındaki fark olarak hesaplanır.

SGD algoritmasında, ağırlık güncellemesi doğrudan her batch sonrası gradient bilgisiyle yapılır. Çıkış ve gizli katman ağırlık güncellemeleri için *deltaWeightOutput* ve *deltaWeightHidden* hesaplanır. Güncelleme adımı, gradientin öğrenme oranı (*learningRate*) ile çarpılması ve ardından mevcut ağırlıkların güncellenmesiyle gerçekleştirilir. Her batch sonrasında, modelin eğitim kaybı (*currentLoss*) hesaplanır. Eğer mevcut kayıp değeri (*currentLoss*), en iyi kayıp değerinden daha düşükse, bu durumda en iyi ağırlıklar *bestWeights* ve en iyi kayıp değeri *bestLoss* olarak güncellenir.

Eğitim süreci tamamlandığında, modelin en iyi ağırlık değerleri ve en düşük kayıp değeri döndürülür. SGD, özellikle büyük veri kümelerinde hızlı bir şekilde optimize etme avantajına sahiptir. Her batch için ayrı ayrı güncelleme yaparak modelin hızla öğrenmesini sağlarken, tam veri kümesini kullanmadığı için düşük bellek tüketir. Bununla birlikte, gradyanların tek örneklere göre güncellenmesi zaman zaman gürültülü güncellemelere yol açabilir, bu da yerel minimumdan kaçınmayı sağlar ancak bu durum bazen öğrenmenin dalgalı olmasına neden olmaktadır.

Adam (Adaptive Moment Estimation)

Tez çalışmasına ait uygulama kısmında kullanılan önemli optimizasyon algoritmalarından olan Adam, makine öğrenimi ve derin öğrenmede yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Gradyanların birinci ve ikinci momentlerinin tahminlerini kullanarak uyarlanabilir öğrenme oranlarını belirler. Bu algoritma hem Momentum hem de RMSprop optimizasyon algoritmalarının ilkelerini birleştirir (X. Liang et al., 2023). Adam optimizasyon algoritması için matematiksel denklemler kümesi sırayla Denklem (12), (13), (14), (15) ve (16)'da gösterilmiştir.

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t \tag{12}$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2 \tag{13}$$

$$\widehat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_2^{t+1}} \tag{14}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}} \tag{15}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{m_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon} \tag{16}$$

Adam optimizer için matematiksel denklemlerdeki parametreler aşağıdaki gibidir: m_t ve v_t sırasıyla birinci ve ikinci momentlerin tahminlerini gösterir. g_t Gradyan vektörüdür. β_1 ve β_2 hem gradyanların ani dalgalanmalarını hem de uzun vadeli eğilimlerini dikkate alarak öğrenme sürecini hızlandıran parametrelerdir. α öğrenme oranıdır. ∈ sıfıra bölme hatalarını önlemek için kullanılan çok küçük bir değerdir. Bu denklemler moment tahminlerini günceller ve ağırlıkları ayarlayarak Adam'ın farklı problemleri ele almasına ve öğrenme oranını otomatik olarak ayarlamasına olanak tanır. Tablo 8'de tez çalışmasında kullanılan Adam yönteminin eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 8. Tez çalışmasında kullanılan Adam yönteminin eğitim aşamasındaki sözde kodu

***Input**: numTrain, epochs, learningRate, initialWeights, beta1, beta2, epsilon*
***Output**: bestWeights, bestLoss*
Begin
weights ← initialWeights
bestWeights ← weights
bestLoss ← ∞
mHidden ← 0, vHidden ← 0
mOutput ← 0, vOutput ← 0
t ← 1
***for** epoch ← 1 to epochs do*
  ***for** batch ← 1 to numTrain do*
    // İleri yayılım
    *hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)*
    *outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)*
    // Hata hesaplama
    error ← Output - outputLayerOutput
    // Hata geri yayılımı
    *outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)*
    *hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) * ActivationDerivative(hiddenLayerOutput)*
    // Ağırlık güncelleme
    *mOutput ← beta1 * mOutput + (1 - beta1) * outputLayerGradient*
    *vOutput ← beta2 * vOutput + (1 - beta2) * outputLayerGradient^2*
    *mHidden ← beta1 * mHidden + (1 - beta1) * hiddenLayerGradient*
    *vHidden ← beta2 * vHidden + (1 - beta2) * hiddenLayerGradient^2*
    mOutputHat ← mOutput / (1 - beta1^t)
    vOutputHat ← vOutput / (1 - beta2^t)
    mHiddenHat ← mHidden / (1 - beta1^t)
    vHiddenHat ← vHidden / (1 - beta2^t)
    *deltaWeightOutput ← -learningRate / sqrt(vOutputHat + epsilon) * mOutputHat*
    *deltaWeightHidden ← -learningRate / sqrt(vHiddenHat + epsilon) * mHiddenHat*
    weights.outputLayer ← weights.outputLayer + deltaWeightOutput
    weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
    t ← t + 1
    // Eğitim kaybını hesapla
    currentLoss ← LossFunction(error)
    ***if** currentLoss < bestLoss then*

```
            bestWeights ← weights
            bestLoss ← currentLoss
        End if
    End for
End for
return bestWeights, bestLoss
End
```

Adam, Momentum ve RMSprop algoritmalarını birleştiren ve genellikle en iyi performansı sunan bir optimizasyon yöntemidir. Adam, önceki gradyanların ortalamasını (momentum) ve RMSprop'taki kare gradient ortalamasını kullanarak hem stabil hem de hızlı bir güncelleme yapar.

Adam, adaptif öğrenme hızına sahip olan ve her parametre için öğrenme hızını ayrı ayrı ayarlayan bir optimizasyon tekniğidir. *Train.cpp* dosyasındaki Adam algoritması, modelin ağırlıklarını *initialWeights* olarak başlatarak en iyi ağırlıklar (*bestWeights*) ile en düşük kayıp değeri (*bestLoss*) ilk başta tanımlanır. İlk momentum (*mHidden*, *mOutput*) ve ikinci momentum (*vHidden*, *vOutput*) sıfırdan başlatılır, *t* ise zaman adımı olarak kullanılan bir sayaç olup başlangıçta sıfırdır.

Eğitim aşaması, belirli bir *epoch* ve *batch* sayısınca tekrarlanır. Her *epoch* kümesinde her bir *batch* için ileri ve geri yayılım gerçekleştirilir. İleri yayılım sırasında, model girdileri gizli katmandan çıkış katmanına doğru ilerleyerek *hiddenLayerOutput* ve *outputLayerOutput* değerlerini oluşturur. Bu değerler, modelin tahmin sonuçları olup, hata hesaplamasında kullanılır. Modelin tahmini ile gerçek çıkış arasındaki fark *error* olarak adlandırılır ve modelin hatasını yansıtır.

Geri yayılım aşamasında, modelin gradyanları hesaplanır. Çıkış katmanındaki gradyan (*outputLayerGradient*), hatanın aktivasyon fonksiyonunun türev değeri ile çarpılmasıyla bulunur. Gizli katmandaki gradyan (*hiddenLayerGradient*), çıkış katmanından gelen gradyanın gizli katmandaki aktivasyon türevi ile çarpılması sonucu elde edilir.

Adam algoritması iki ayrı momentum güncellemesi yapar: *m* ilk momentum olarak gradyanın hareketli ortalamasını (*momentum*) ve *v* ise gradyanın karesinin hareketli ortalamasını temsil eder. İlk momentum (*mOutput* ve *mHidden*), *beta1* katsayısı ile güncellenir; bu katsayı, önceki momentumun mevcut gradientle ne kadar harmanlanacağını belirler. İkinci momentum (*vOutput* ve *vHidden*), *beta2* katsayısı kullanılarak güncellenir ve gradyan karelerinin bir ortalamasını oluşturur.

Daha sonra, her iki momentum da kaydırılmış ortalama (*bias correction*) hesaplanarak güncellenir. Bu adım, özellikle başlangıçtaki momentumların etkisini azaltarak dengeleme yapar ve gradyanın doğru bir tahminini sağlar. Bu kaydırılmış ortalama değerler kullanılarak *deltaWeightOutput* ve *deltaWeightHidden* güncellemeleri yapılır. Her iki katman için ağırlıklar, gradyanın karesine dayalı bir normalizasyon ile öğrenme hızı adaptif hale getirilerek güncellenir.

Her *batch* sonrası, modelin *currentLoss* değeri hesaplanır. Eğer bu kayıp değeri *bestLoss* değerinden daha düşükse, en iyi ağırlıklar ve kayıp değeri güncellenir. Bu süreç, en düşük kayıpla en iyi ağırlıkları elde etmeyi sağlar. Eğitim tamamlandığında, *bestWeights* ve *bestLoss* değeri geri döndürülür.

RMSprop (Root Mean Square Propagation)

RMSprop (Ma et al., 2023) AdaGrad'ı modifiye eden bir algoritmadır. RMSprop, AdaGrad'da karşılaşılan azalan öğrenme oranları sorununu ele almak için geliştirilmiştir. Bu optimizasyon tekniği, bireysel parametre öğrenme oranlarını uyarlamalı olarak ayarlamak için karesel gradyanların çalışan ortalamasını kullanır. RMSprop optimizasyon algoritması için matematiksel Denklem 17'de sunulmuştur.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t}+\epsilon} \odot g_t \tag{17}$$

RMSprop optimizasyon algoritmasının matematiksel denklemindeki parametreler aşağıdaki gibidir: θ_t: t adımındaki model parametreleri, g_t: t adımındaki gradyan, η: öğrenme oranı, v_t: karesel gradyanların üstel ağırlıklı hareketli ortalaması, ∈: sıfıra bölme hatalarını önlemek için çok küçük bir sayı ve ⊙: eleman bazında çarpma. Bu denklemde her parametre için ayrı bir öğrenme oranı hesaplanır. Eski gradyanların karelerinin çalışan ortalaması, mevcut gradyanın karesi ile güncellenir. Bu, öğrenme oranlarını gradyanların büyüklüklerine göre dinamik olarak ayarlar. Tablo 9'da tez çalışmasında kullanılan RMSProp algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir

Tablo 9. Tez çalışmasında kullanılan RMSProp yönteminin eğitim aşamasındaki sözde kodu

***Input**: numTrain, epochs, learningRate, initialWeights, gamma, epsilon*
***Output**: bestWeights, bestLoss*
Begin
1: *weights ← initialWeights*
2: *bestWeights ← weights*
3: *bestLoss ← ∞*
4: *gradSquarePrevHidden ← 0*

```
 5:  gradSquarePrevOutput ← 0
 6:  for epoch ← 1 to epochs do
 7:     for batch ← 1 to numTrain do
 8:        // İleri yayılım
 9:        hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
10:        outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
11:        // Hata hesaplama
12:        error ← Output - outputLayerOutput
13:        // Hata geri yayılımı
14:        outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
15:        hiddenLayerGradient ← (weights.outputLayer^T*outputLayerGradient)*
                                 ActivationDerivative(hiddenLayerOutput)
16:        // Ağırlık güncelleme (RMSprop)
17:       gradSquarePrevOutput ← gamma * gradSquarePrevOutput + (1 - gamma) * outputLayerGradient^2
18:      gradSquarePrevHidden ← gamma * gradSquarePrevHidden + (1 - gamma) * hiddenLayerGradient^2
19:        deltaWeightOutput ← -learningRate / sqrt(gradSquarePrevOutput + epsilon) *
                                                 outputLayerGradient * hiddenLayerOutput^T
20:        deltaWeightHidden ← -learningRate / sqrt(gradSquarePrevHidden + epsilon) *
                                                    hiddenLayerGradient * Input^T
21:        weights.outputLayer ← weights.outputLayer + deltaWeightOutput
22:        weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
23:        // Eğitim kaybını hesapla
24:        currentLoss ← LossFunction(error)
25:        if currentLoss < bestLoss then
26:           bestWeights ← weights
27:           bestLoss ← currentLoss
28:        End if
29:     End for
30:  End for
31:  return bestWeights, bestLoss
End
```

RMSProp (Root Mean Square Propagation) algoritması, gradyanın her parametre için adaptif bir öğrenme oranı ile güncellenmesini sağlayarak optimize eder. *Train.cpp* dosyasındaki RMSProp algoritması, modelin ağırlıklarını *initialWeights* ile başlatarak başlangıçta *bestWeights* ve *bestLoss* değişkenlerini tanımlanır. *bestWeights*, her adımda en iyi model ağırlıklarını saklarken *bestLoss*, en düşük kayıp değeri olarak güncellenir. İlk kare gradyan ortalamaları (*gradSquareHidden* ve *gradSquareOutput*) sıfır olarak tanımlanır.

Eğitim döngüsü, belirli bir epoch sayısı boyunca devam eder. Her *epoch*, belirli sayıda *batch* üzerinde çalışır. İleri yayılım sırasında, modelin girdisi gizli katmandan çıkış katmanına doğru geçer ve *hiddenLayerOutput* ve *outputLayerOutput* hesaplanır. Bu çıktılar modelin tahmin sonuçlarıdır ve gerçek çıktılarla karşılaştırılarak hata (*error*) hesaplanır.

Hata geri yayılımı aşamasında, bu hata değeri gradyan hesaplamasında kullanılır. Çıkış katmanındaki gradyan (*outputLayerGradient*), modelin hatasının aktivasyon fonksiyonunun

türev değeri ile çarpılmasıyla elde edilir. Gizli katman gradyanı (*hiddenLayerGradient*), çıkış katmanından gelen gradyan ile gizli katmandaki aktivasyon türevi kullanılarak hesaplanır.

RMSProp algoritmasında, her gradyanın karesi alınır ve hareketli ortalama ile güncellenir. Kare gradyanların ortalaması (*gradSquareOutput* ve *gradSquareHidden*), gamma (γ) katsayısı ile eski ortalama ve (*1-gamma*) katsayısı ile yeni gradyan karesi harmanlanarak hesaplanır. Bu hareketli ortalama, öğrenme hızının parametreler üstündeki etkisini normalleştirmek için kullanılır. Her bir parametre için adaptif olarak güncellenen bu öğrenme hızı, gradyanların büyüklüğüne göre öğrenme oranını azaltır ve büyük gradyanların parametreler üzerindeki etkisini dengeler.

Son olarak, *deltaWeightOutput* ve *deltaWeightHidden* ağırlık güncellemeleri adaptif olarak öğrenme hızına göre güncellenir. Bu güncellemeler, *RMSProp'un* epsilon (ε) sabitini kullanarak gradyanın karesi üzerindeki bölmeyi stabilize eder ve öğrenme oranının sıfıra yaklaşmasını engeller. Yeni ağırlıklar, bu güncellemelerle güncellenir ve her batch sonrası modelin kaybı (*currentLoss*) hesaplanır. Eğer *currentLoss* değeri *bestLoss* değerinden daha düşükse, *bestWeights* ve *bestLoss* güncellenir.

Eğitim süreci sona erdiğinde, *bestWeights* ve *bestLoss* döndürülür. RMSProp algoritması, özellikle gürültülü ve yüksek boyutlu veri setlerinde stabil öğrenme sağlar. Bu adaptif yöntem, gradyanların büyüklüğüne göre öğrenme hızını dinamik olarak ayarlayarak yakınsamanın daha dengeli ve hızlı gerçekleşmesini sağlar. Bu sayede, derin öğrenme modellerinde ve karmaşık ağlarda daha istikrarlı bir performans sergiler.

Adagrad (Adaptive Gradient Algorithm)

AdaGrad, bireysel parametrelerin öğrenme oranlarını son gradyanlarına göre optimize eder (Z. Cai et al., 2023). Büyük türevlere sahip parametreler öğrenme hızında hızlı bir düşüş yaşarken, küçük türevlere sahip olanlar daha az önemli ölçüde azalır. Bu, gradyanların geçmiş karesel değerleri kullanılarak yapılır. Özellikle seyrek veriler veya farklı ölçeklerde özelliklere sahip veriler için kullanışlıdır. AdaGrad optimizasyon algoritması için matematiksel Denklem 18'de gösterilmektedir.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}+\epsilon}} \cdot g_{t,i} \tag{18}$$

AdaGrad optimize edici için matematiksel denklemlerdeki parametreler aşağıdaki gibidir: $\theta_{t+1,i}$: t+1 adımındaki ağırlık parametresi, $\theta_{t,i}$: t adımındaki ağırlık parametresi, η: öğrenme

oranı, $G_{t,ii}$: t adımındaki i parametresi için önceki gradyanların karelerinin toplamı, $g_{t,i}$: t adımındaki i parametresi için gradyan değeri ve $\in$: sıfıra bölme hatalarını önlemek için çok küçük bir değer. AdaGrad denkleminde her parametre için ayrı bir öğrenme oranı hesaplanır. Geçmiş gradyanların karelerinin toplamı mevcut gradyan ile güncellenir. Böylece, öğrenme oranı seyrek güncellenen parametreler için artarken, sık güncellenenler için azalır. Tablo 10'da çalışmada yer verilen AdaGrad algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 10. Çalışmada yer verilen AdaGrad algoritmasının eğitim aşamasındaki sözde kodu

```
Input: numTrain, epochs, learningRate, initialWeights, epsilon
Output: bestWeights, bestLoss
Begin
 1: weights ← initialWeights
 2: bestWeights ← weights
 3: bestLoss ← ∞
 4: gradSquareSumHidden ← 0
 5: gradSquareSumOutput ← 0
 6: for epoch ← 1 to epochs do
 7:    for batch ← 1 to numTrain do
 8:       // İleri yayılım
 9:       hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
10:       outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
11:       // Hata hesaplama
12:       error ← Output - outputLayerOutput
13:       // Hata geri yayılımı
14:       outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
15:       hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) *
                                ActivationDerivative(hiddenLayerOutput)
16:       // Ağırlık güncelleme (Adagrad)
17:       gradSquareSumOutput ← gradSquareSumOutput + outputLayerGradient^2
18:       gradSquareSumHidden ← gradSquareSumHidden + hiddenLayerGradient^2
19:       deltaWeightOutput ← -learningRate / sqrt(gradSquareSumOutput + epsilon) *
                                                    outputLayerGradient * hiddenLayerOutput^T
20:       deltaWeightHidden ← -learningRate / sqrt(gradSquareSumHidden + epsilon) *
                                                    hiddenLayerGradient * Input^T
21:       weights.outputLayer ← weights.outputLayer + deltaWeightOutput
22:       weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
23:       // Eğitim kaybını hesapla
24:       currentLoss ← LossFunction(error)
25:       if currentLoss < bestLoss then
26:          bestWeights ← weights
27:          bestLoss ← currentLoss
28:       End if
29:    End for
30: End for
31: return bestWeights, bestLoss
End
```

Adagrad (Adaptive Gradient Algorithm) algoritması, parametrelerin geçmişteki tüm gradyanlarının karelerini toplayarak, her parametre için adaptif bir öğrenme oranı belirler. *Train.cp*p dosyasındaki Adagrad algoritması, modelin başlangıç ağırlıklarını *initialWeights* ile başlatır ve en iyi ağırlıklar (*bestWeights*) ile en düşük kayıp değeri (*bestLoss*) ilk başta tanımlanır. İlk kare gradyan toplamları (*gradSquareSumHidden* ve *gradSquareSumOutput*) sıfır olarak başlatılır.

Eğitim aşaması, belirli bir *epoch* ve *batch* sayısınca tekrarlanır. Her *epoch* kapsamında her bir *batch* için ileri ve geri yayılım işlemleri gerçekleştirilir. İleri yayılım sırasında, model girdisi gizli katmandan çıkış katmanına doğru geçerek *hiddenLayerOutput* ve *outputLayerOutput* değerlerini hesaplar. Modelin tahmin ettiği *outputLayerOutput*, gerçek çıkış (*Output*) ile karşılaştırılarak hata (*error*) hesaplanır.

Hata geri yayılım aşamasında, bu hata gradient hesaplaması için kullanılır. Çıkış katmanındaki gradyan (*outputLayerGradient*), hatanın aktivasyon fonksiyonunun türev değeri ile çarpılmasıyla elde edilirken, gizli katmandaki gradyan (*hiddenLayerGradient*), çıkış katmanından gelen gradyan ile gizli katmandaki aktivasyon türevinin çarpılmasıyla hesaplanır.

Adagrad algoritmasının karakteristik özelliği, her bir gradyanın karesinin zamanla birikerek toplanmasıdır. *gradSquareSumOutput* ve *gradSquareSumHidden* değişkenlerinde, her bir gradyanın karesi, toplam kare gradyanların hareketli bir ortalamasını oluşturacak şekilde birikir. Bu toplam, her parametreye özel adaptif bir öğrenme oranı oluşturmak için kullanılır.

Ağırlık güncelleme adımında, her bir parametre, öğrenme hızının gradyanın karelerinin toplamının karekökü ile bölünmesiyle güncellenir. Böylece, sık güncellenen parametrelerin öğrenme oranı azalırken nadiren güncellenen parametrelerin öğrenme oranı daha yüksek kalır. Epsilon (ε) sabiti, karekök işlemi sırasında sıfır bölme hatasını engellemek için eklenir.

Her *batch* sonrası, modelin *currentLoss* değeri hesaplanır. Eğer bu kayıp değeri, *bestLoss* değerinden daha düşükse, en iyi ağırlıklar (*bestWeights*) ve en düşük kayıp değeri (*bestLoss*) güncellenir. Bu süreç, modelin eğitimi tamamlanana kadar devam eder ve en iyi ağırlıklar döndürülür. Bu tez çalışmasında AdaGrad yönteminin memristör tabanlı uygulamalarda verimli bir yöntem olduğu ortaya çıkmıştır.

AdaDelta

AdaDelta, AdaGrad'ın azalan öğrenme oranları sorununu çözmek için geliştirilmiş bir uyarlamasıdır (Zeiler, 2012). Bu algoritma, sürekli azalan öğrenme oranlarına ve global bir

öğrenme oranı seçmeye olan ihtiyacı ortadan kaldırır. AdaDelta, sabit bir pencere boyutunu korurken önceki tüm gradyanlara eşit önem vererek, geçmiş gradyanlardan oluşan kayan bir pencere kullanarak karesel gradyanların hareketli ortalamasını hesaplar. AdaDelta optimizasyon algoritması için matematiksel denklemler Denklem (19), (20), (21), (22) ve (23)'te gösterilmiştir.

$$RMS\,[E[g^2]]_t = \sqrt{E[g^2]_t + \epsilon} \tag{19}$$

$$RMS\,[\Delta x^2]_{t-1} = \sqrt{E[\Delta x^2]_{t-1} + \epsilon} \tag{20}$$

$$update = -\frac{RMS\,[\Delta x]_{t-1}}{RMS\,[E[g^2]]_t} g_t \tag{21}$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1-\rho) update^2 \tag{22}$$

$$x_{t+1} = x_t + update \tag{23}$$

Tablo 11'de çalışmada kullanılan AdaDelta algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 11. Çalışmada kullanılan AdaDelta algoritmasının eğitim aşamasındaki sözde kodu

```
Input: numTrain, epochs, initialWeights, gamma, epsilon
Output: bestWeights, bestLoss
Begin
 1: weights ← initialWeights
 2: bestWeights ← weights
 3: bestLoss ← ∞
 4: gradSquarePrevHidden ← 0
 5: gradSquarePrevOutput ← 0
 6: deltaPrevHidden ← 0
 7: deltaPrevOutput ← 0
 8: for epoch ← 1 to epochs do
 9:     for batch ← 1 to numTrain do
10:         // İleri yayılım
11:         hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
12:         outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
13:         // Hata hesaplama
14:         error ← Output - outputLayerOutput
15:         // Hata geri yayılımı
16:         outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
17:         hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient)
                                  * ActivationDerivative(hiddenLayerOutput)
18:         // Ağırlık güncelleme (Adadelta)
19:         gradSquarePrevOutput ← gamma * gradSquarePrevOutput + (1 - gamma) * outputLayerGradient^2
20:         gradSquarePrevHidden ← gamma * gradSquarePrevHidden + (1 - gamma) * hiddenLayerGradient^2
21:         deltaWeightOutput ← - sqrt((deltaPrevOutput + epsilon) / (gradSquarePrevOutput + epsilon)) *
                                  outputLayerGradient * hiddenLayerOutput^T
22:         deltaWeightHidden ← - sqrt((deltaPrevHidden + epsilon) / (gradSquarePrevHidden + epsilon)) *
                                  hiddenLayerGradient * Input^T
23:         deltaPrevOutput ← gamma * deltaPrevOutput + (1 - gamma) * deltaWeightOutput^2
```

```
24:        deltaPrevHidden ← gamma * deltaPrevHidden + (1 - gamma) * deltaWeightHidden^2
25:        weights.outputLayer ← weights.outputLayer + deltaWeightOutput
26:        weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
27:        // Eğitim kaybını hesapla
28:        currentLoss ← LossFunction(error)
29:        if currentLoss < bestLoss then
30:           bestWeights ← weights
31:           bestLoss ← currentLoss
32:        End if
33:     End for
34:  End for
35:  return bestWeights, bestLoss
End
```

Tablo 11'de gösterilen sözde kodda görüldüğü gibi, Adadelta optimizasyon algoritmasını kullanarak bir modelin ağırlıklarını günceller ve en iyi ağırlık değerlerini kaydeder.

Tez çalışması uygulama kısmında eğitim süreci başlamadan önce, *weights* yani modelin ağırlıkları, başlangıç değerleri olan *initialWeights* ile ayarlanır. *bestWeights* değişkeni de bu başlangıç ağırlıklarıyla başlatılır ve ilerleyen adımlarda en iyi ağırlıklar burada saklanır. *bestLoss* başlangıçta sonsuz *(∞)* olarak tanımlanır, böylece ilk hesaplanan kayıptan küçük her değer, *bestLoss* olarak güncellenir. Ayrıca, Adadelta algoritmasının bileşenleri olan *gradSquarePrevHidden*, *gradSquarePrevOutput*, *deltaPrevHidden* ve *deltaPrevOutput* sıfırdan başlatılır. Bu değişkenler, hareketli ortalamalar için önceki adımların bilgisini saklar. *for* döngüsü ile *epoch* sayısı kadar eğitim iterasyonu yapılır. Her *epoch*, tüm eğitim verileri üzerinden bir geçişi temsil eder. Her *epoch* içinde, *for* döngüsüyle her bir *batch* (veri kümesinin bir alt kümesi) üzerinde işlem yapılır. İleri yayılım (f*orward-propagation*) aşamasında, modelin tahminleri hesaplanır. İlk olarak, giriş verisi (*input*) ile gizli katmanın (*hidden layer*) ağırlıkları çarpılır ve *hiddenLayerOutput* elde edilir. Daha sonra, *hiddenLayerOutput* ile çıktı katmanının (*output layer*) ağırlıkları çarpılır ve *outputLayerOutput* hesaplanır. Bu işlemler, aktivasyon fonksiyonları kullanılarak gerçekleştirilir. Çıkış (*Output*) ve *outputLayerOutput* arasındaki fark *error* olarak tanımlanır. Bu, modelin tahmin ettiği değer ile gerçek değer arasındaki farktır. Hata geri yayılımı (*Backward Propagation*) ile *outputLayerGradient* ve *hiddenLayerGradient* hesaplanır. *outputLayerGradient*, çıkış katmanındaki hatanın gradyanını temsil eder. *hiddenLayerGradient* ise, çıkış katmanından geri yayılmak üzere gizli katmandaki hatanın gradyanını içerir. Aktivasyon fonksiyonunun türevini kullanarak bu gradyanlar hesaplanır. Adadelta optimizasyon algoritması, her iki katmanın gradyanlarını kullanarak ağırlıkları günceller. *gradSquarePrevOutput* ve *gradSquarePrevHidden*, önceki kare gradyan ortalamalarının ağırlıklı toplamı olarak güncellenir. *gamma* faktörü ile eski değeri, *(1 - gamma)*

faktörü ile de yeni gradyan karesi eklenir. Böylece, her yeni *batch'*te ağırlıklandırılmış bir hareketli ortalama oluşturulur. *deltaWeightOutput* ve *deltaWeightHidden*, ağırlık değişimi için kullanılan güncellemeleri içerir. Bu değerler, Adadelta'nın adaptif öğrenme hızı kullanılarak hesaplanır. Güncelleme oranı, önceki kare ağırlık değişimleri ve gradyan ortalamaları arasındaki oranla belirlenir. Bu oran, her bir gradyan için normalize edilmiş bir değer sağlar ve güncelleme miktarının büyük veya küçük olmasını önler. *deltaPrevOutput* ve *deltaPrevHidden*, yeni ağırlık değişimi karelerini saklar ve güncellenen değerlerle hareketli ortalamalarını sürdürür. *weights.outputLayer* ve *weights.hiddenLayer*, Adadelta algoritmasıyla elde edilen güncelleme değerleri (*deltaWeightOutput* ve *deltaWeightHidden*) kullanılarak güncellenir. *currentLoss*, *LossFunction* ile *error* kullanılarak hesaplanır ve bu, modelin mevcut batch üzerindeki performansını gösterir. Bu kayıp değeri, modelin doğruluğunu değerlendirmek için kullanılır. Eğer *currentLoss*, *bestLoss* değerinden küçükse, *bestWeights* ve *bestLoss* güncellenir. Bu işlem, modelin en iyi ağırlık ve en düşük kayıp değerlerinin kaydedilmesini sağlar. Eğitim döngüsünün tamamlanmasının ardından, modelin en iyi ağırlıkları (*bestWeights*) ve en düşük kayıp değeri (*bestLoss*) geri döndürülür.

Tez çalışmasında kullanılan, Adadelta algoritmasının eğitim sürecinde adaptif güncellemeleri nasıl yaptığını ve en iyi model parametrelerini nasıl kaydettiğini özetlemektedir.

Nadam (Nesterov-Accelerated Adaptive Moment Estimation)

Nadam, Adam'ın güncelleme kuralına Nesterov momentumunu ekler. Bu yöntem, momentum yönünde gelecekteki bir konumu dikkate alarak gradyanı hesaplar. Böylece yakınsama hızını artırmayı ve model kalitesini iyileştirmeyi amaçlar. Nadam optimizasyon algoritması için matematiksel denklemler Denklem (24), (25), (26), (27) ve (28)'de gösterilmiştir.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{24}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{25}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_2^t} \tag{26}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{27}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t}+\epsilon} (\beta_1 \widehat{m}_t + (1-\beta_1) g_t \tag{28}$$

Nadam optimizasyon algoritmasının matematiksel denklemlerindeki parametreler aşağıdaki gibidir: m_t ve v_t : sırasıyla t adımdaki birinci ve ikinci momentlerin tahminleri. g_t: gradyan vektörü. β_1 ve β_2 : hareketli ortalamalar için üstel bozunma oranları. η: öğrenme oranı.

$\widehat{m}_t$ ve $\hat{v}_t$: birinci ve ikinci momentlerin düzeltilmiş tahminleri. ∈: sıfıra bölme hatalarını önlemek için kullanılan çok küçük bir sayı. Nadam, önceki momentum kavramını geliştirmek için Nesterov Momentum ve Adam algoritmalarını birleştirir ve böylece yerel minimumlara takılma riskini azaltır. Tablo 12'de çalışmada yer verilen Nadam algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 12. Çalışmada kullanılan Nadam algoritmasının eğitim aşamasındaki sözde kodu

```
Input: numTrain, epochs, learningRate, initialWeights, beta1, beta2, epsilon
Output: bestWeights, bestLoss
Begin
 1:  weights ← initialWeights
 2:  bestWeights ← weights
 3:  bestLoss ← ∞
 4:  mHidden ← 0
 5:  vHidden ← 0
 6:  mOutput ← 0
 7:  vOutput ← 0
 8:  t ← 0
 9:  for epoch ← 1 to epochs do
10:     for batch ← 1 to numTrain do
11:         t ← t + 1
12:         // İleri yayılım
13:         hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
14:         outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
15:         // Hata hesaplama
16:         error ← Output - outputLayerOutput
17:         // Hata geri yayılımı
18:         outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
19:         hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) *
                                  ActivationDerivative(hiddenLayerOutput)
20:         // Ağırlık güncelleme (Nadam)
21:         mOutput ← beta1 * mOutput + (1 - beta1) * outputLayerGradient
22:         vOutput ← beta2 * vOutput + (1 - beta2) * outputLayerGradient^2
23:         mHidden ← beta1 * mHidden + (1 - beta1) * hiddenLayerGradient
24:         vHidden ← beta2 * vHidden + (1 - beta2) * hiddenLayerGradient^2
25:         mOutputHat ← mOutput / (1 - beta1^t)
26:         vOutputHat ← vOutput / (1 - beta2^t)
27:         mHiddenHat ← mHidden / (1 - beta1^t)
28:         vHiddenHat ← vHidden / (1 - beta2^t)
29:         deltaWeightOutput ← -learningRate * (beta1 * mOutputHat + (1 - beta1) * outputLayerGradient) /
                                (sqrt(vOutputHat) + epsilon) * hiddenLayerOutput^T
30:         deltaWeightHidden ← -learningRate * (beta1 * mHiddenHat + (1 - beta1) *
                                hiddenLayerGradient) / (sqrt(vHiddenHat) + epsilon) * Input^T
31:         weights.outputLayer ← weights.outputLayer + deltaWeightOutput
32:         weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
33:         // Eğitim kaybını hesapla
34:         currentLoss ← LossFunction(error)
35:         if currentLoss < bestLoss then
36:             bestWeights ← weights
```

```
            bestLoss ← currentLoss
        End if
    End for
End for
return bestWeights, bestLoss
End
```

Nadam optimizasyon yöntemi, Adam algoritmasının bir uzantısı olup, adaptif öğrenme hızını Nesterov momentumu ile birleştirir. Bu yöntemde, gradientlerin momentum etkisi altında hızlandırılması sağlanırken, her adımda daha iyi sonuç almak için Nesterov momentumu kullanılır. Uygulamada ağın eğitiminin gerçekleştirildiği *Train.cpp* dosyasındaki Nadam algoritmasında öncelikle modelin ağırlıkları *initialWeights* ile başlatılır ve en iyi ağırlıklar *bestWeights* olarak kaydedilir. En iyi kayıp değeri başlangıçta ∞ olarak ayarlanır, böylece eğitim boyunca ilk güncellenen kayıp değeri en düşük olarak kabul edilir ve kaydedilir. Momentum ve ikinci momentum olarak kullanılan *mHidden*, *vHidden*, *mOutput* ve *vOutput* sıfırdan başlatılır. *t* değişkeni ise zaman adımını takip etmek için sıfırdan başlatılır.

Eğitim sürecinde, belirlenen *epoch* sayısı boyunca model güncellenir. Her *epoch* içinde tüm *batch* verileri işlenir. İleri yayılım sırasında modelin giriş katmanından başlayarak hesaplamalar yapılır. Giriş verisi gizli katmanın ağırlıklarıyla çarpılarak *hiddenLayerOutput* hesaplanır, ardından bu değer çıkış katmanına aktarılır ve modelin tahmini olan *outputLayerOutput* elde edilir. Tahmin değeri ile gerçek değer arasındaki fark *error* olarak hesaplanır, bu hata değeri modelin o anki performansını yansıtır.

Geri yayılım sırasında, hatanın her katmana geri iletilmesiyle gradyanlar hesaplanır. Çıkış katmanı için *outputLayerGradient*, hata ile aktivasyon fonksiyonunun türev değeri çarpılarak elde edilir. Gizli katmandaki gradyan ise, çıkış katmanından geriye doğru hesaplanan gradyanın, gizli katmandaki aktivasyon türevi ile çarpılmasıyla bulunur. Bu gradyanlar, ağırlık güncellemeleri için kullanılacaktır.

Nadam optimizasyonunda, önce ilk momentum ve ikinci momentum değerleri güncellenir. *mOutput* ve *mHidden*, mevcut gradyanlarla güncellenerek ilk momentum değerini oluşturur ve *vOutput* ile *vHidden* da gradyanların kareleri üzerinden ikinci momentum olarak güncellenir. Momentum terimleri kaydırılmış *(bias-corrected)* ortalamalar olarak hesaplanır; böylece algoritma başlangıçtaki dengesizliklerin etkisinden kurtulur ve daha hassas sonuçlar elde edilir.

Ağırlık güncellemeleri yapılırken, Nesterov momentumu ile gelecekteki bir adımda hareket göz önünde bulundurulur. *deltaWeightOutput* ve *deltaWeightHidden*, kaydırılmış ortalama ve karekök ile normalize edilen gradyanlar kullanılarak hesaplanır. Bu güncellemeler

adaptif öğrenme hızını baz alarak modelin daha verimli bir şekilde en düşük kayba ulaşmasını sağlar.

Son olarak, modelin performansı her *batch* sonrasında hesaplanır. Eğer modelin ilgili *batch* üzerindeki kayıp değeri en düşük değer olarak güncellenirse, ağırlıklar *bestWeights* olarak saklanır. Eğitim süreci tamamlandığında, modelin en iyi ağırlık ve kayıp değerleri döndürülür. Nadam yöntemi, gradyanların hareketlerini gelecekteki adımlar üzerinden optimize ederek daha hızlı ve kararlı bir öğrenme sağlar, bu da modelin minimum kayba daha etkin bir biçimde ulaşmasına yardımcı olur.

Momentum

Momentum yöntemi (Ruder, 2016) fiziksel momentum kavramına benzemektedir. Amaç, önceki adımların hızını bir momentum terimi ile hesaba katarak optimizasyon yönünde daha hızlı hareket sağlamaktır. Bu yaklaşımla, önceki gradyanlardan elde edilen veriler bir *"momentum"* etkisiyle korunarak her seferinde daha hızlı güncelleme yapılmasına olanak sağlanır. Momentum optimizasyon algoritması için matematiksel denklemler Denklem (29) ve (30)'te gösterilmektedir.

$$v_{t+1} = \beta_1 v_t + \alpha g_t \tag{29}$$

$$\theta_{t+1} = \beta_1 v_t + \alpha g_t \tag{30}$$

Momentum optimizasyon algoritmasının matematiksel denklemlerindeki parametreler aşağıdaki gibidir. v_{t+1}: momentumu temsil eder ve geçmiş gradyanların ağırlıklı bir ortalamasıdır. θ_t: t adımındaki model parametreleri. g_t: t adımındaki gradyan. α: öğrenme oranı. β_1: momentumu kontrol eden bir hiperparametredir. Tablo 13'te çalışmada yer verilen momentum algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 13. Çalışmada kullanılan Momentum algoritmasının eğitim aşamasındaki sözde kodu

```
Input: numTrain, epochs, learningRate, initialWeights, gamma
Output: bestWeights, bestLoss
Begin
 1: weights ← initialWeights
 2: bestWeights ← weights
 3: bestLoss ← ∞
 4: velocityHidden ← 0
 5: velocityOutput ← 0
 6: for epoch ← 1 to epochs do
 7:     for batch ← 1 to numTrain do
 8:         // İleri yayılım
 9:         hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
10:         outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
```

```
        // Hata hesaplama
        error ← Output - outputLayerOutput
        // Hata geri yayılımı
        outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
        hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) *
                                    ActivationDerivative(hiddenLayerOutput)
        // Ağırlık güncelleme (Momentum)
        velocityOutput ← gamma * velocityOutput - learningRate * outputLayerGradient *
                                                              hiddenLayerOutput^T
        velocityHidden ← gamma * velocityHidden - learningRate * hiddenLayerGradient * Input^T
        weights.outputLayer ← weights.outputLayer + velocityOutput
        weights.hiddenLayer ← weights.hiddenLayer + velocityHidden
        // Eğitim kaybını hesapla
        currentLoss ← LossFunction(error)
        if currentLoss < bestLoss then
            bestWeights ← weights
            bestLoss ← currentLoss
        End if
    End for
End for
return bestWeights, bestLoss
End
```

Momentum algoritması, SGD'ye göre daha kararlı bir güncelleme yapmayı amaçlar. Momentum, bir yöne doğru hızlanmayı sağlar ve bu sayede minimum noktaya daha çabuk ve stabil bir biçimde ulaşılır. SGD'de, gradyan değeri her örneğe göre dalgalandığından ağırlıkların güncellenmesi de gürültülüdür. Momentum, önceki adımların ortalamasını alarak daha düzgün bir güncelleme yapılmasını sağlar. Sözde kodda önce ağırlıklar ve hız vektörleri sıfırdan başlatılır. Her *epoch* ve *batch* için ileri yayılım ve hata hesaplaması yapılır, ardından hata geri yayılımıyla gradyan değerleri bulunur. Momentum aşamasında, önceki gradyan adımları da dikkate alınarak ağırlıkların güncellenmesi sağlanır. Bu sayede yerel minimumlara takılmadan, daha hızlı bir yakınsama elde edilir. Momentum katsayısı (gamma), önceki gradyan adımlarının ağırlıklarını belirler; bu katsayı daha büyük olduğunda güncellemeler daha stabil hale gelir. Eğer mevcut kayıp değeri, daha önceki en iyi kayıp değerinden düşükse, bu değer en iyi sonuç olarak saklanır. Aşağıda uygulamada kullanılan Momentum optimizasyon yöntemi detaylı bir biçimde açıklanmıştır.

Başlangıç değerleri eğitim başlamadan önce, *weights* değişkeni modelin başlangıç ağırlık değerleri olan *initialWeights* ile başlatılır. *bestWeights*, başlangıçta weights olarak ayarlanır ve süreç boyunca en iyi ağırlık değerlerini kaydeder. *bestLoss* ise ∞ (sonsuz) olarak belirlenir, böylece elde edilen ilk kayıp değeri *bestLoss*'dan daha düşük olacaktır ve en iyi kayıp değeri olarak güncellenir. Momentum yöntemi için ayrıca *velocityHidden* ve *velocityOutput*

değişkenleri sıfırdan başlatılır; bu değişkenler, ağırlık güncellemelerinin hızını temsil eder ve momentum etkisini içerir. Epoch döngüsü ile belirlenen *epochs* sayısı kadar çalışır ve her *epoch*, tüm eğitim çıktıları üzerinden bir tam geçişi ifade eder. Batch döngüsü her *epoch* içinde, *batch* döngüsü ile her bir *batch* üzerinde eğitim işlemi yapılır. Model her *batch'te* güncellenir ve böylece gradyanlar daha sık bir şekilde güncellemeye katkı sağlar. İleri Yayılım (Forward Propagation), modelin giriş katmanından çıkış katmanına kadar veri akışı gerçekleşir. Giriş verisi, gizli katmanın ağırlıkları ile çarpılarak *hiddenLayerOutput* elde edilir. Ardından, *hiddenLayerOutput* çıkış katmanının ağırlıklarıyla çarpılır ve *outputLayerOutput* hesaplanır. Bu işlemde, aktivasyon fonksiyonu kullanılarak katmanların çıktıları hesaplanır. Hata hesaplama ile çıkış (*Output*) ile modelin tahmini *(outputLayerOutput)* arasındaki fark *error* olarak hesaplanır. Bu hata değeri, modelin o anki *batch'te* ne kadar sapma gösterdiğini ifade eder. Hata Geri Yayılımı (Backward Propagation), modeldeki hatalar, katmanlarda geriye doğru iletilir. Çıkış katmanı için *outputLayerGradient*, hata (*error*) ile aktivasyon fonksiyonunun türev değerinin çarpımıyla hesaplanır. Gizli katman için *hiddenLayerGradient*, çıkış katmanından geri yayılım ile elde edilen gradyanın, gizli katman aktivasyonunun türevi ile çarpılması sonucu hesaplanır.

Momentum yöntemi kullanılarak güncellemeler, önceki adımlarda hesaplanan gradyan bilgisinin bir kısmı korunarak *velocityHidden* ve *velocityOutput* güncellemeleri yapılır. Bu güncellemeler, *gamma* faktörüyle önceki *velocity* değerini çarparak momentum etkisini sağlar. *learningRate* ile çarpılan güncel gradient ise eklenir. Böylece *velocity*, güncel *gradientin* hareketini yumuşatır ve geçmiş *gradientlerin* etkisini de barındırır. Ağırlıkların güncellenmesi, elde edilen *velocityOutput* ve *velocityHidden* değerleri, *weights.outputLayer* ve *weights.hiddenLayer* ağırlıklarına eklenerek ağırlıklar güncellenir. Bu işlem, modelin mevcut *batch* üzerindeki gradyan yönünde ilerlemesini sağlar, ancak momentum yöntemi sayesinde bu hareket daha kararlı ve istikrarlıdır. Eğitim kaybının hesaplanması (*LossFunction)*, *error* kullanılarak *currentLoss* hesaplanır. Bu değer, modelin mevcut *batch* üzerindeki kayıp miktarını gösterir ve modelin o anki performansını değerlendirir. En iyi ağırlık ve kayıp güncellemesinde eğer *currentLoss*, *bestLoss* değerinden düşükse, *bestWeights* ve *bestLoss* güncellenir. Bu, modelin şimdiye kadarki en iyi performans gösterdiği ağırlıkların ve kayıp değerinin saklanmasını sağlar. Sonuç olarak eğitim sürecinin sonunda, *bestWeights* ve *bestLoss* değerleri döndürülür. Bu değerler, modelin en iyi performansı gösterdiği ağırlıkları ve en düşük kaybı içerir.

Momentum yöntemi, modelin minimuma daha süratli ve daha kararlı bir biçimde ulaşmasına yardımcı olur. Ağırlık güncellemelerinde, gradientlerin geçici dalgalanmalarına

karşı bir dengeleme sağlar ve geçmişteki gradyan hareketlerinin etkisiyle güncellemeler yapılır. Bu sayede, model yerel minimumlardan kaçınarak daha stabil bir optimizasyon süreci gerçekleştirir.

Adamax

Adam'ın bir varyantı olarak geliştirilen AdaMax, sonsuzluk normunu kullanarak yakınsama kararlılığını iyileştirmeyi amaçlamaktadır (Long et al., 2023). Bu yaklaşımda, gradyanların üstel hareketli ortalamalarının L-sonsuzluk normu hesaplanır ve Adam algoritmasındaki gradyanın sonsuzluk normu yerine kullanılır. AdaMax optimizasyon algoritması için matematiksel denklemler Denklem (19), (20) ve (21)'de gösterilmiştir.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{31}$$

$$u_t = max(\beta_2 u_{t-1,} \|g_t\|_\infty \tag{32}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{1-\beta_1^t} \frac{m_t}{u_t + \epsilon} \tag{33}$$

AdaMax optimizasyon algoritmasının matematiksel denklemlerindeki parametreler aşağıdaki gibidir. θ_t: *t* adımındaki model parametreleri. g_t: t adımındaki gradyan. η: öğrenme oranı. β_1 ve β_2 hareketli ortalamalar için üstel bozunma oranlarını temsil eden iki parametre. m_t: *t* adımındaki ilk moment tahmini. u_t: t adımdaki ilk moment tahmini ve sonsuzluk normundan (maksimum mutlak değer) seçilen maksimum değer. ∈: sıfıra bölme hatalarını önlemek için kullanılan çok küçük bir sayı. AdaMax, Adam optimizasyon algoritmasına benzer şekilde ikinci moment yerine sonsuzluk normunu kullanarak daha istikrarlı bir performans sağlamayı amaçlamaktadır. Bu durumun sonucu olarak, özellikle derin ağlarda daha iyi sonuçların alınmasını sağlayabilir. Tablo 14'te Adamax algoritmasının eğitim aşamasındaki sözde kodu gösterilmiştir.

Tablo 14. Adamax algoritmasının eğitim aşamasındaki sözde kodu

Input: *numTrain, epochs, learningRate, initialWeights, beta1, beta2, epsilon*
Output: *bestWeights, bestLoss*
Begin
1: weights ← initialWeights
2: bestWeights ← weights
3: bestLoss ← ∞
4: mHidden ← 0
5: uHidden ← 0
6: mOutput ← 0
7: uOutput ← 0
8: t ← 0
9: ***for*** *epoch ← 1 to epochs do*

```
    for batch ← 1 to numTrain do
        t ← t + 1
        // İleri yayılım
        hiddenLayerOutput ← ActivationFunction(weights.hiddenLayer * Input)
        outputLayerOutput ← ActivationFunction(weights.outputLayer * hiddenLayerOutput)
        // Hata hesaplama
        error ← Output - outputLayerOutput
        // Hata geri yayılımı
        outputLayerGradient ← error * ActivationDerivative(outputLayerOutput)
        hiddenLayerGradient ← (weights.outputLayer^T * outputLayerGradient) *
                                    ActivationDerivative(hiddenLayerOutput)
        // Ağırlık güncelleme (Adamax)
        mOutput ← beta1 * mOutput + (1 - beta1) * outputLayerGradient
        uOutput ← max(beta2 * uOutput, |outputLayerGradient|)
        mHidden ← beta1 * mHidden + (1 - beta1) * hiddenLayerGradient
        uHidden ← max(beta2 * uHidden, |hiddenLayerGradient|)
        mOutputHat ← mOutput / (1 - beta1^t)
        mHiddenHat ← mHidden / (1 - beta1^t)
        deltaWeightOutput ← -learningRate * mOutputHat / (uOutput + epsilon) * hiddenLayerOutput^T
        deltaWeightHidden ← -learningRate * mHiddenHat / (uHidden + epsilon) * Input^T
        weights.outputLayer ← weights.outputLayer + deltaWeightOutput
        weights.hiddenLayer ← weights.hiddenLayer + deltaWeightHidden
        // Eğitim kaybını hesapla
        currentLoss ← LossFunction(error)
        if currentLoss < bestLoss then
            bestWeights ← weights
            bestLoss ← currentLoss
        End if
    End for
End for
return bestWeights, bestLoss
End
```

Adamax, ikinci momentumun *(v)* güncellenmesinde $L\infty$ normunu kullanır ve bu sayede, gradyanın büyüklüğünden daha az etkilenir ve daha dengeli güncellemeler sağlar. Adamax algoritması, öncelikle modelin başlangıç ağırlıkları (*initialWeights*) ile başlar ve en iyi ağırlıklar (*bestWeights*) ile en düşük kayıp (*bestLoss*) değeri başlangıçta tanımlanır. İlk momentum (*mHidden*, *mOutput*) ve $L\infty$ normuna göre güncellenen ikinci momentum (*uHidden*, *uOutput*) sıfırdan başlatılır. *t* ise zaman adımı olarak kullanılan bir sayaçtır ve başlangıçta sıfırdır.

Eğitim süreci, belirli bir *epoch* ve *batch* sayısınca tekrarlanır. Her *epoch* dahilinde her bir *batch* için ileri yayılım gerçekleştirilir. Modelin girdileri, gizli katmandan çıkış katmanına doğru ilerleyerek *hiddenLayerOutput* ve *outputLayerOutput* değerlerini oluşturur. Bu değerler, modelin tahmin sonuçları olup, hata hesaplamasında kullanılır. Modelin tahmini ile gerçek çıkış arasındaki fark *error* olarak adlandırılır ve modelin hatasını yansıtır.

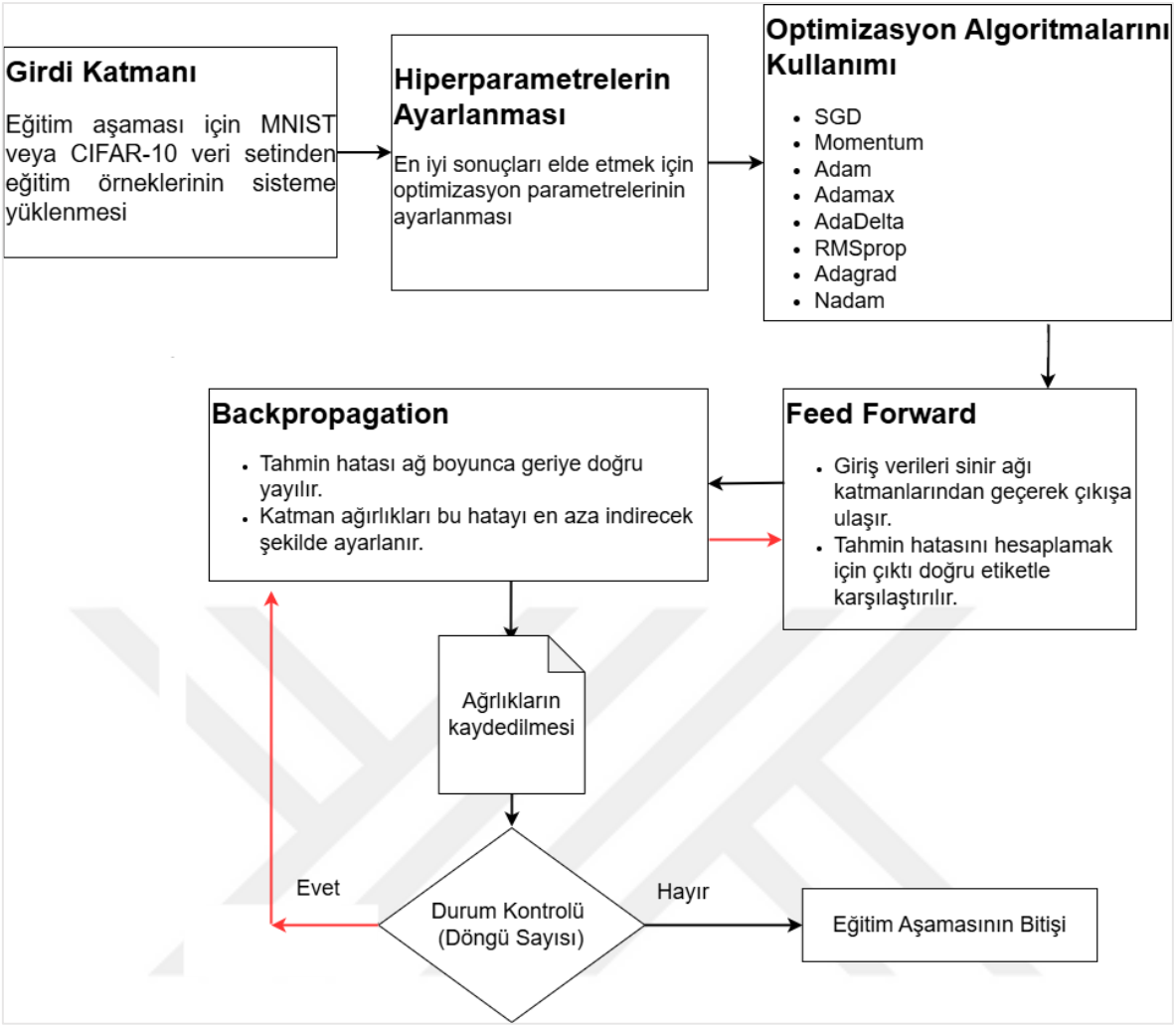
Adamax algoritmasında, ilk momentum (*m*) ve *L*∞ normuna göre ikinci momentum (*u*) güncellenir. İlk momentum (*mOutput* ve *mHidden*), beta1 katsayısı ile mevcut gradientlerin hareketli ortalaması olarak hesaplanır. İkinci momentum ise *uOutput* ve *uHidden* değişkenlerinde *beta2* katsayısı kullanılarak her bir gradientin *L*∞ normuna göre güncellenir, yani mevcut gradient ile *u*'nun *beta2* katsayısına göre ölçeklendirilmiş eski değeri arasında maksimum olanı seçilir. Bu işlem, büyük gradient değişimlerinden daha az etkilenilmesini sağlar ve daha kararlı bir güncelleme sunar.

İlk momentum, *mOutput* ve *mHidden* için kaydırılmış ortalama (*bias correction*) hesaplanır; bu sayede başlangıçtaki dengesizliklerin etkisi azaltılmış olur. Daha sonra ağırlık güncellemeleri yapılır. Çıkış ve gizli katman ağırlıkları için, kaydırılmış ortalamaya göre güncellenen momentum değerleri (*mOutputHat*, *mHiddenHat*) ve *L*∞ normu (*uOutput*, *uHidden*) ile normalize edilerek *deltaWeightOutput* ve *deltaWeightHidden* hesaplanır. Bu adımlar, adaptif bir öğrenme hızında güncelleme yapılmasını sağlar.

Her batch sonrasında eğitim kaybı hesaplanır ve eğer bu kayıp değeri mevcut en iyi kayıptan daha düşükse, *bestWeights* ve *bestLoss* güncellenir. Bu, modelin eğitim süresince en düşük kayıpla en iyi ağırlıkları öğrenmesini sağlar.

Eğitim tamamlandığında, modelin en iyi ağırlıkları (*bestWeights*) ve kayıp değeri (*bestLoss*) döndürülür. Adamax algoritması, *L*∞ normu kullanarak gradyanların büyüklüğünden daha az etkilenir ve bu sayede büyük gradyan değişimlerine karşı daha kararlıdır. Bu özellikleriyle, büyük veri kümelerinde Adamax, öğrenme sürecinde daha istikrarlı bir ilerleme sağlar. Yapılan tez çalışmasında AdaMax optimizasyon yöntemi ile başarılı sonuçlara ulaşılmıştır.

Şekil 29'de uygulamaya ait eğitim aşaması belirtilmiştir. Bu şekil, tez çalışmasında kullanılan uygulamaya ait yapay sinir ağının eğitim sürecini adım adım açıklayan bir akış diyagramıdır.



Şekil 29. Uygulamanın eğitim aşaması.

Uygulamanın test aşaması

Bu tez çalışmasında, test kısmının çalışma mantığını açıklamak amacıyla donanım ve yazılım bölümlerine ait sözde kodlar Tablo 15 ve Tablo 16'da sunulmuştur. Uygulamada kullanılan MNIST ve CIFAR-10 veri setlerinde eğitim için 50.000 bin örnek, test aşaması için ise 10.000 örnek resim objesi kullanılmaktadır. Ayrıca uygulamanın sağlamlığını pekiştirmek amacıyla Fisher's Iris veri seti uygulamaya dahil edilmiştir. Burada ileri beslemeli bir sinir ağında (*feedforward neural network*) test verilerini kullanarak sınıflandırma işlemi gerçekleştirmektir. Program hem donanım hem de yazılım kısmında çeşitli optimizasyonlar ve hesaplamalar yaparak enerji sarfiyatını minimum düzeye indirmeyi hedefler. Kodun genel akışı üç ana bileşenden oluşur: donanım parametrelerinin başlatılması, tüketilen enerjinin hesaplanması ve ileri besleme işleminin gerçekleştirilmesidir.

Donanım aşaması

Tablo 15. Uygulamanın test kısmına ait sözde kod

```
Input: arrayIH, arrayHO, param, techIH, techHO, testInput, weight1, weight2
Output: correct, sumArrayReadEnergyIH, sumArrayReadEnergyHO
Begin
1:  Function InitializeHardwareParameters():
2:     // Donanım parametrelerini başlat
3:     if arrayIH->cell[0][0] type-> "eNVM" then:
4:        readVoltageIH ← arrayIH->cell[0][0].readVoltage
5:        readVoltageHO ← arrayHO->cell[0][0].readVoltage
6:        readPulseWidthIH ← arrayIH->cell[0][0].readPulseWidth
7:        readPulseWidthHO ← arrayHO->cell[0][0].readPulseWidth
9:     if arrayIH->cell[0][0] type-> "HybridCell" then:
10:       readVoltageIH ← arrayIH->cell[0][0].LSB->readVoltage
11:       readVoltageHO ← arrayHO->cell[0][0].LSB->readVoltage
12:       readVoltageMSB ← arrayIH->cell[0][0].MSB->readVoltage
13:       readPulseWidthIH ← arrayIH->cell[0][0].LSB->readPulseWidth
14:       readPulseWidthHO ← arrayHO->cell[0][0].LSB->readPulseWidth
15:       readPulseWidthMSB ← arrayIH->cell[0][0].MSB->readPulseWidth
16: End Function
18: Function CalculateEnergyConsumption():
19:    sumArrayReadEnergyIH ← 0
20:    sumArrayReadEnergyHO ← 0
22:    // Giriş Katmanı İçin Enerji Tüketimi Hesapla
23:    Her bir gizli katman sinapsı için:
24:       if arrayIH->cell type-> "AnalogNVM" then:
25:          if arrayIH->cell CMOS then:
26:             sumArrayReadEnergyIH += arrayIH->wireGateCapRow * (techIH.vdd ^ 2) * param->nInput
27:          else, arrayIH crosbarr then:
28:             sumArrayReadEnergyIH += arrayIH->wireCapRow * (techIH.vdd ^ 2) * (param->nInput - 1)
30:       if arrayIH->cell type-> "DigitalNVM" then:
31:          if arrayIH->cell CMOS then:
32:             sumArrayReadEnergyIH += arrayIH->wireGateCapRow * (techIH.vdd ^ 2)
33:          else arrayIH crossbar ise:
34:             sumArrayReadEnergyIH += arrayIH->wireCapRow * (techIH.vdd ^ 2) * (param->nInput - 1)
36:       if arrayIH->cell type-> "HybridCell" then:
37:          sumArrayReadEnergyIH += arrayIH->wireGateCapRow * (techIH.vdd ^ 2) * param->nInput
39:    // Çıkış Katmanı İçin Enerji Tüketimi Hesapla
40:    return sumArrayReadEnergyHO
41: End Function
End
```

Uygulamaya ait sözde kodun ilk aşamasında, donanım parametrelerinin yapılandırılması işlemi gerçekleştirilir. Bu işlem, sinir ağı modelinin donanımda çalıştırılmasını sağlayacak parametrelerin başlatılmasını içerir. *InitializeHardwareParameters()* fonksiyonu, her hücre türüne *(cell type)* göre okuma voltajı *(readVoltage)* ve okuma darbe genişliği *(readPulseWidth)*

gibi parametreleri ayarlar. Kod, donanım katmanında kullanılan hücre türünü *(eNVM, HybridCell)* tespit ederek uygun voltaj ve darbe genişliği değerlerini belirler.

- Eğer hücre tipi "*eNVM*" (*Embedded Non-Volatile Memory*) ise, okuma işlemi için kullanılan voltaj ve darbe genişliği doğrudan *arrayIH* ve *arrayHO* nesnelerinden alınır.
- Eğer hücre tipi "HybridCell" ise, hücrelerin alt bileşenleri olan LSB (*Least Significant Bit*) ve MSB (*Most Significant Bit*) için okuma voltajları ve darbe genişlikleri ayrı ayrı ayarlanır. Bu durum, hibrit hücrelerin farklı özelliklere sahip alt bileşenleri kullanarak daha esnek bir okuma işlemi gerçekleştirmesini sağlar.

Test aşmasının bu kısmında donanım tabanlı bir YSA modelinin temel gereksinimlerini oluşturarak enerji tüketimi ve performans hesaplamalarına geçiş yapılmasını sağlar.

Enerji tüketiminin hesaplanmasında *CalculateEnergyConsumption()* fonksiyonu kullanılmaktadır. Bu işlem, sinir ağı modelinin ileri besleme aşamasında donanım kaynaklarının etkin şekilde kullanılması için önemlidir. Enerji tüketimi, sinapslar ve bağlantılar üzerinden geçen akımın voltaj ve kapasitans (*capacitance*) değerlerine bağlı olarak hesaplanır.

- Giriş Katmanı Enerji Tüketimi: Kod, *AnalogNVM*, *DigitalNVM* ve *HybridCell* gibi farklı hücre türlerine göre giriş katmanındaki enerji tüketimini hesaplar. Eğer hücreler *CMOS* tabanlı ise, *wireGateCapRow* değeri ile voltaj (*techIH.vdd*) karesi çarpılır ve *param->nInput* ile ölçeklenir. Aksi takdirde, çapraz bağlantılı yapılar için *wireCapRow* değeri kullanılır.
- Çıkış Katmanı Enerji Tüketimi: Çıkış katmanında da benzer işlemler tekrarlanır, ancak bu defa *arrayHO* nesnesi üzerinden enerji tüketimi hesaplanır.

Bu enerji hesaplamaları, sinir ağının çalışırken harcadığı enerjiyi en aza indirmek ve güç verim seviyesini yükseltmek için kritik bir rol oynar. Sinapsların her biri için yapılan bu hesaplamalar, toplam enerji tüketimini *sumArrayReadEnergyIH* ve *sumArrayReadEnergyHO* değişkenlerinde saklar.

Yazılım aşaması

Tablo 16. Uygulamanın test kısmına ait sözde kodun devamı (yazılım)

```
42: Function ForwardPropagationUsingSoftware():
43:     correct ← 0
45:     Tüm test görüntüleri için:
46:         // Gizli katman başlangıç değerlerini sıfırla
47:         outN1[i] ← 0
48:         a1[i] ← 0
49:         // Çıkış katman başlangıç değerlerini sıfırla
```

```
50:       outN2[i] ← 0
51:       a2[i] ← 0
53:       // Giriş Katmanından Gizli Katmana İleri Besleme
54:       if param->useHardwareInTestingFF Then:
55:          // Donanımda enerji tüketimi hesapla
56:          CalculateEnergyConsumption()
57:       else:
58:          Her bir gizli katman sinapsı için:
59:             // Giriş ile sinaps ağırlığını çarp ve toplama ekle
60:             outN1[j] += testInput[i][k] * weight1[k][j]
61:          a1[j] ← Activate(outN1[j])
63:       // Gizli Katmandan Çıkış Katmanına İleri Besleme
64:       Her bir çıkış katman sinapsı için:
65:          // Gizli katman aktivasyon çıkışlarını çarp ve toplama ekle
66:          outN2[m] += a1[j] * weight2[j][m]
67:       a2[m] ← Activate(outN2[m])  // Aktivasyon fonksiyonu uygula
69:       if a2 then:
70:          correct += 1
71: End Function
73: Function ReportTestResults():
74:    accuracy = correct / param->numMnistTestImages
75:    totalEnergyConsumption = sumArrayReadEnergyIH + sumArrayReadEnergyHO
76:    delay = CalculateDelay()
77:    result (accuracy, totalEnergyConsumption, delay)
78: End Function
```

Yazılım Tabanlı İleri Besleme, sinir ağı modelinin ileri besleme aşaması, yazılım tabanlı olarak *ForwardPropagationUsingSoftware()* fonksiyonu ile gerçekleştirilir. Bu işlem, modelin eğitim sırasında öğrenilen ağırlıkları kullanarak test verilerini sınıflandırmasını sağlar. Bu aşamada *correct* değişkeni, doğru tahminlerin sayısını kaydetmek için kullanılır.

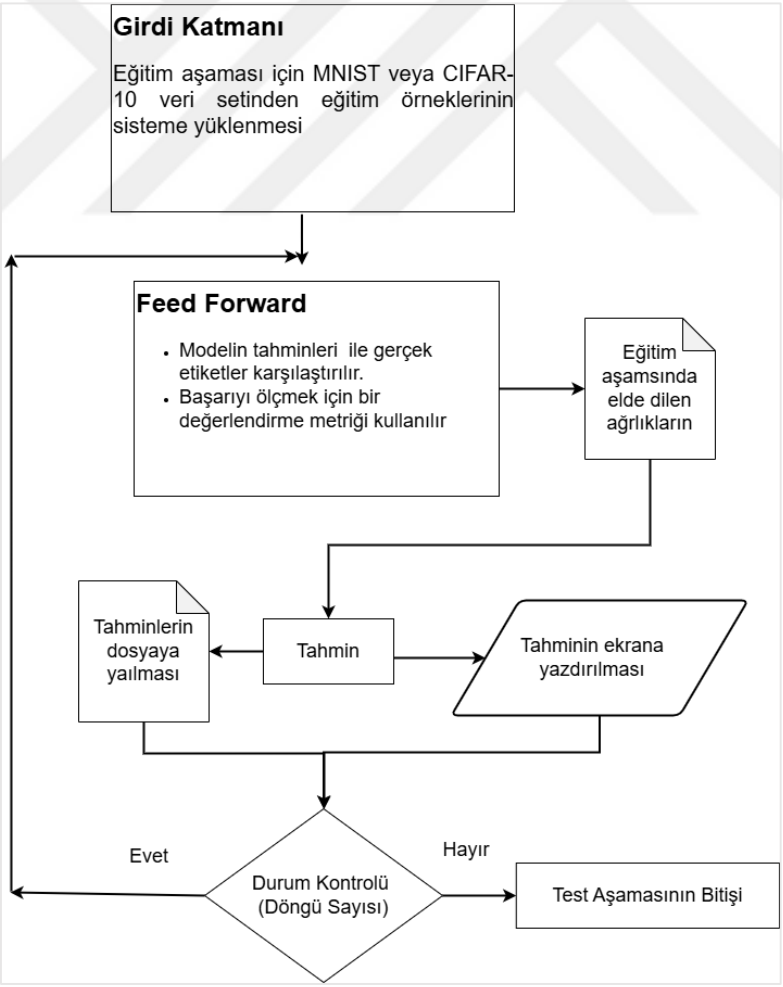
- Gizli Katmana İleri Besleme: Giriş katmanındaki sinirler ile gizli katmandaki sinirler arasında çarpımlar yapılarak gizli katman çıkışları (*outN1* ve *a1*) hesaplanır. Eğer *param>useHardwareInTestingFF* parametresi etkinse, ileri besleme sırasında enerji tüketimi donanımda hesaplanır. Aksi takdirde, yazılım tabanlı ileri besleme algoritması çalıştırılır.
- Çıkış Katmanına İleri Besleme: Gizli katmandan çıkış katmanına doğru aynı işlem adımları tekrar edilir. Gizli katman sinirlerinin aktivasyon çıkışları *(a1)*, çıkış katmanı sinir ağırlıkları ile çarpılır ve çıkış değerleri *(outN2 ve a2)* elde edilir.

Son aşamada, *ReportTestResults()* fonksiyonu, sinir ağı modelinin doğruluk oranını *(accuracy)*, toplam enerji tüketimini *(totalEnergyConsumption)* ve gecikme süresini *(delay)* hesaplar. Bu değerler, modelin donanım tabanlı bir ortamda nasıl performans gösterdiğini ve ne kadar enerji tükettiğini analiz etmek için önemlidir.

- Doğruluk Oranı (*Accuracy*): Modelin doğru sınıflandırdığı test verilerinin toplam test verilerine oranı olarak hesaplanır.
- Toplam Enerji Tüketimi (*Total Energy Consumption*): Giriş ve çıkış katmanlarındaki enerji tüketimlerinin toplamı olarak hesaplanır.
- Gecikme Süresi (*Delay*): İleri besleme sürecinde gerçekleşen işlemler için toplam gecikme süresi hesaplanır.

Sonuçlar, modelin performansını ve güç verimliliğini değerlendirerek daha verimli donanım ve yazılım kombinasyonları tasarlamak için kullanılır. Bu raporlanan değerler, donanım tabanlı sinir ağları için güç verimliliği optimizasyonu ve performans artırımı açısından önemli bir referans sağlar.

Şekil 30'de uygulamaya ait test aşaması gösterilmiştir. Bu şekil, tez çalışmasında kullanılan uygulamaya ait yapay sinir ağının test sürecini adım adım açıklayan bir akış diyagramıdır.



Şekil 30. Uygulamanın test aşaması.

Test aşaması, yapay sinir ağının eğitim sırasında öğrendiği bilgilerin yeni, daha önce görülmemiş veriler üzerinde değerlendirilmesi sürecidir. Bu aşamada, ağın performansını ölçmek ve genelleme yeteneğini değerlendirmek amaçlanır. Test verileri, eğitim sırasında kullanılan veri setinden farklıdır ve aynı ön işleme adımlarından geçirilmiştir. Ağ, test verilerini ileri yayılım (forward propagation) ile işler ve bu veriler için tahminler üretir.

Elde edilen tahminler, test veri setindeki gerçek etiketlerle karşılaştırılır. Performans, doğruluk (accuracy), hataların ortalaması (MSE, MAE gibi) veya F1 skor gibi uygun metrikler ile değerlendirilir. Bu aşamada ağın ağırlıkları değiştirilmez, yalnızca eğitim sırasında öğrenilen bilgilerin yeni verilere nasıl uygulandığı gözlemlenir. Sonuçlar, modelin genelleme kapasitesini ölçmek ve gerekirse iyileştirmeler yapmak için kullanılır. Test aşaması, eğitim sürecinin doğruluğunu ve etkinliğini değerlendirmenin en kritik adımlarından biridir.

Uygulamanın çalıştırılması

Bu tez çalışmasında donanım destekli yapay sinir ağı (YSA) modellerinin verimli bir biçimde eğitim ve değerlendirme süreçlerini ele almaktadır. Geleneksel YSA eğitim algoritmaları, büyük veri kümeleri üzerinde hesaplama açısından yoğun işlemler gerektirdiğinden, yüksek enerji tüketimi ve uzun işlem süreleri gibi sınırlamalarla karşı karşıya kalmaktadır. Bu sorunları aşmak için, bu çalışma YSA'nın donanımsal bir platformda eğitilmesi amacıyla bir yöntem geliştirmekte ve simülasyon tabanlı bir sinaptik çekirdek yapılandırması kullanmaktadır. Kod, eğitim verilerinin işlenmesi, sinaptik ağırlıkların başlatılması, NeuroSim çekirdeklerinin yapılandırılması ve YSA modelinin donanım üzerinde doğruluk, gecikme süresi ve enerji tüketimi gibi performans metriklerinin hesaplanması adımlarını içermektedir. Bu yaklaşımla, YSA'nın donanımda etkin şekilde çalıştırılması sağlanarak hem enerji verimliliği hem de işlem süresi açısından iyileştirmeler hedeflenmektedir. Tablo 17'de ana fonksiyon kullanılarak veri setlerinin yüklenmesi ve çeşitli fonksiyonlarla uygulamanın başlatılması gösterilmiştir.

Tablo 17. Uygulamnın çalıştırıldığı ana fonksiyona ait sözde kod

***Input**: trainData, trainLabels, testData, testLabels, totalNumEpochs, interNumEpochs, numMnistTestImages, numTrainImagesPerEpoch, optimization_type*
***Output**: output.csv, accuracy and performance metrics*
Begin
1: ***Initialize random seed and load MNIST data***
2: *ReadTrainingDataFromFile("trainData", "trainLabels")*
3: *ReadTestingDataFromFile("testData", "testLabels")*
4: ***Initialize synaptic arrays and NeuroSim cores***

```
5:  arrayIH->Initialization<RealDevice>()
6:  arrayHO->Initialization<RealDevice>()
7:  NeuroSimSubArrayInitialize and NeuroSimNeuronInitialize
8:  Calculate area and leakage power
9:  NeuroSimSubArrayArea and NeuroSimSubArrayLeakagePower
10: Initialize weights and hardware mapping
11: WeightInitialize() and WeightToConductance() (if useHardwareInTraining)
12: Open output file output.csv
13: For each epoch block
14:  Run training for interNumEpochs
15:  If not using hardware in training but in testing, then WeightToConductance()
16:  Run validation
17:  Save accuracy to file and print metrics
18:  Print read/write latency and energy for synaptic cores
19:  If HybridCell or _2T1F, print transfer latency and energy
20: End for
21: Return 0
End
```

Tablo 17'de gösterilen sözde kodda, uygulamada bir MLP modelini donanımsal platformda eğitim ve test süreçlerini özetlemektedir. Kod, eğitim verilerini yükler, modelin donanım tarafını kurar, ağı eğitir ve sonunda doğrulama yapar. Süreç aşağıdaki aşamalardan oluşmaktadır:

- Rastgele Sayı Çekirdeği ve Veri Yükleme: Kod, rastgele sayı üretimini başlatmak için bir çekirdek değeri ayarlar ve MNIST, CIFAR-10 veya Fisher's Iris veri setinden eğitim ve test verilerini yükler. Bu adım, modelin eğitim ve test aşamalarında kullanılacak görüntü ve etiketlerin bellek içine alınmasını sağlar.
- Sinaptik Ağırlık Dizilerinin ve *NeuroSim* Çekirdeklerinin Başlatılması: YSA'nın gizli katmanına ve çıktı katmanına ait sinaptik ağırlık dizileri, `*RealDevice*` kullanılarak başlatılır. *NeuroSim* çekirdeklerinin ayarları yapılır ve bu sinaptik çekirdekler, belirli donanım özelliklerine göre yapılandırılır.
- Alan ve Kaçak Güç Hesaplama: Modelin donanımsal yapısını simüle eden NeuroSim çekirdeklerinin alan ve *standby* (bekleme) modunda enerji tüketimleri hesaplanır. Bu hesaplamalar, modelin donanım üzerinde kapladığı fiziksel alanı ve düşük güç modundayken harcadığı enerjiyi tahmin etmeye yardımcı olur.
- Ağırlıkların Başlatılması ve Donanıma Uygun Şekilde Haritalanması: Modelin sinaptik ağırlıkları başlatılır ve donanım üzerinde çalışacak şekilde iletkenlik değerlerine dönüştürülür. Eğer eğitimde donanım kullanılacaksa, bu dönüştürme işlemi yapılır.

- Çıktı Dosyasının Açılması: Eğitim süreci boyunca modelin doğruluk oranlarının kaydedileceği *`output.csv`* dosyası açılır.
- Eğitim ve Doğrulama Döngüsü: Kod, *`totalNumEpochs`* boyunca modeli eğitir ve her *`interNumEpochs`* sonrası doğrulama yapar. Eğitim süreci tamamlandığında doğruluk oranı hesaplanır ve `output.csv` dosyasına yazılır.
- Performans ve Enerji Tüketim Metriği Hesaplama ve Yazdırma: Eğitim ve doğrulama aşamaları boyunca sinaptik çekirdeklerin okuma/yazma gecikme süreleri ile enerji tüketimleri hesaplanır ve çıktı olarak yazdırılır. Eğer *"HybridCell"* veya *"_2T1F"* gibi özel hücre tipleri kullanılıyorsa, aktarım gecikme süreleri ve enerji tüketimleri de ayrıca rapor edilir.
- Sonuçların Çıktısı ve Döndürme: Eğitim süreci tamamlandığında, modelin doğruluk oranları ve donanım üzerindeki performans metrikleri "*output.cs"* dosyasına yazılmış olur ve '*main*' fonksiyonu başarıyla sonlanır.

Performans ölçüm metrikleri

Donanım tabanlı öğrenme sistemlerinin performansını değerlendirmek için kullanılan temel ölçütler; doğruluk, test hata oranı, kesinlik ve özgüllüktür. Doğruluk (*accuracy*), sistemin yaptığı doğru tahminlerin toplam tahminlere oranını ifade eder ve genel model başarısının bir göstergesi olarak kullanılır. Test hata oranı (test error rate) ise yanlış sınıflandırmaların oranını temsil eder ve modelin tahminlerindeki hataları değerlendirmek için kritik bir ölçüttür. Bu oran, modelin genelleme yeteneğini ölçmek için önemli bir referans sağlar.

Kesinlik (*precision*), modelin pozitif olarak sınıflandırdığı örneklerin ne kadarının gerçekte doğru olduğunu ölçer. Bu, özellikle yanlış pozitif sınıflandırmaların maliyetli olduğu durumlarda (örneğin, tıbbi teşhis sistemlerinde) oldukça önemli bir metrik olarak karşımıza çıkar. Diğer bir ifadeyle, kesinlik, modelin "gerçek pozitif" olarak adlandırılan doğru pozitif tahminleri tespit etme kapasitesine odaklanır.

Özgüllük (*specificity*) ise modelin negatif sınıflandırmalarındaki başarısını, yani "gerçek negatifleri" doğru bir şekilde tanımlama yeteneğini ölçer. Bu metrik, özellikle yanlış negatiflerin kritik sonuçlar doğurabileceği senaryolarda (örneğin, güvenlik sistemlerinde) modelin doğruluk seviyesini analiz etmek için kullanılır.

Bu ölçütler birlikte değerlendirildiğinde, donanım tabanlı öğrenme sistemlerinin hem doğruluk hem de hata toleransı açısından ne kadar etkili olduğunu kapsamlı bir şekilde analiz etmeye olanak tanır. Bu çalışma özelinde bu ölçütler bir arada değerlendirilerek uygulanmıştır.

Performans değerlendirmesi sırasında, bu metrikler arasındaki denge de dikkate alınmalı; özellikle bir metrikteki iyileşmenin diğer metrikler üzerindeki potansiyel olumsuz etkileri göz önünde bulundurulmalıdır. Böylece, uygulamaya özgü gereksinimlere uygun optimizasyon stratejileri geliştirilebilir. Bu çalışma, optimizasyon algoritmalarının öğrenme süreçlerini değerlendirmek amacıyla, her yöntem için doğruluk metriğinin hesaplanmasıyla başlamaktadır. Doğruluk *(accuracy)*, bir modelin doğru tahmin oranını temsil etmesi bakımından, model performansını değerlendirmek için kritik bir göstergedir. Sadece tek başına bir metrik olarak değil, aynı zamanda diğer performans ölçütleri ile birlikte ele alındığında, modelin genel etkinliğinin kapsamlı bir değerlendirilmesine olanak tanır. Bu bağlamda, doğruluk metriği, yalnızca modelin genel başarımını göstermekle kalmaz; aynı zamanda veri kümeleri arasında model performansı hakkında ayrıntılı içgörüler sunar. Bu içgörüler, farklı optimizasyon algoritmalarının etkinliklerini karşılaştırma açısından önemli bir temel oluşturur. Özellikle, algoritmaların veri kümesine özgü öğrenme dinamikleri üzerindeki etkilerini anlamak ve modelin genelleme yeteneğini analiz etmek için doğruluk metriği, diğer ölçütlerle entegre bir biçimde kullanılabilir. Ayrıca, bu metriklerin sağladığı karşılaştırmalı analiz, optimizasyon algoritmalarının avantajlı ve dezavantajlı yönlerinin tespit edilmesine olanak tanır. Bu süreç, yalnızca mevcut modellerin daha optimum bir düzeye çıkarılması için değil, aynı zamanda daha verimli ve daha doğru modellerin tasarlanması için de yol gösterici bir rol oynar. Çalışma, doğruluk metriğini, optimizasyon algoritmalarını daha geniş bir performans perspektifinden değerlendirmek için bir temel taş olarak ele almakta ve bu sayede modern makine öğrenimi sistemlerinin performansını artırmaya yönelik değerli katkılar sağlamaktadır. Aşağıda bu metrikler için matematiksel denklemler verilmiştir:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{34}$$

$$Error\ Rate = \frac{FP+FN}{TP+TN+FP+FN} \tag{35}$$

$$Sensitivity = \frac{TP}{TP+FN} \tag{36}$$

$$Specificity = \frac{TN}{TN+FP} \tag{37}$$

Bu metriklerin hesaplanmasında kullanılan terimler aşağıdaki gibi tanımlanmıştır: TP (doğru pozitif), MNIST el yazısı rakamların veya CIFAR-10 veri setindeki resimlerin doğru tanındığı durumların sayısını ifade eder. FP (yanlış pozitif), rakamların doğru olarak yanlış tanındığı durumların sayısını gösterir. TN (doğru negatif) yanlış rakamların doğru şekilde yanlış olarak tanımlandığı vaka sayısını ifade eder. FN (yanlış negatif), doğru olarak tanınması gereken

rakamların yanlış tanımlandığı durumların sayısını temsil eder. Bu terimler, modelin tahmin performansını ayrıntılı olarak analiz etmek için kullanılır ve her birinin kendine özgü anlamı ve önemi vardır.

ARAŞTIRMA BULGULARI VE TARTIŞMA

Deneysel çalışmalar sürecinde, sistemin istikrarlı bir şekilde çalışmasını sağlamak amacıyla uzun süren ve detaylı denemeler gerçekleştirilmiş, bu denemeler sırasında ortaya çıkan çalışma zamanı hataları titizlikle düzeltilmiştir. Sistemin tüm bileşenleri uyum içinde çalışır hale getirildikten sonra, araştırma hipotezlerini test etmek ve araştırma sorularına yanıt bulmak amacıyla çeşitli deneyler planlanmış ve uygulanmaya başlanmıştır. Elde edilen bulgular, değerlendirilip sonuçlara ulaşılmasını takiben, karşılaştırmalı tablolar ve detaylandırılmış grafikler ile görselleştirilmiştir. Sonuçlar, çalışmanın sağlamlığını ve tutarlılığını artıracak şekilde sistematik olarak raporlanmıştır.

Bu tez çalışması, memristör tabanlı nano-sinaptik cihazların kullanıldığı bir sinir ağının donanım tabanlı uygulamasını göstermektedir. Çalışmada işlem süresi, alan gereksinimi ve enerji tüketimi gibi performans sonuçlarını değerlendirilmektedir. Bu hesaplamalar ve deneylerden elde edilen sonuçlar, Tablo 18'de belirtilen özelliklere sahip çalışma ortamında gerçekleştirilmiş ve ölçümleri tamamlanmıştır. Bu çalışma ortamı, deneylerin doğruluğunu ve tekrarlanabilirliğini elde etmek için gerekli koşullar altında oluşturulmuş olup, elde edilen bilgilerin sağlamlığı bu spesifik ortamda yapılan ölçümlerle garanti altına alınmıştır.

Tablo 18. Çalışma Ortamına Ait Özellikler

Birim	**Özellik**
İşlemci	HexaCore Intel Core i7-10750H, 4533 MHz (46 x 99)
Bellek	24 GB 3200 MHz DDR4
Ekran Kartı	nVIDIA GeForce GTX 1650 Ti (HP)
Ön Bellek	12 MB
Hafıza	512 GB SSD

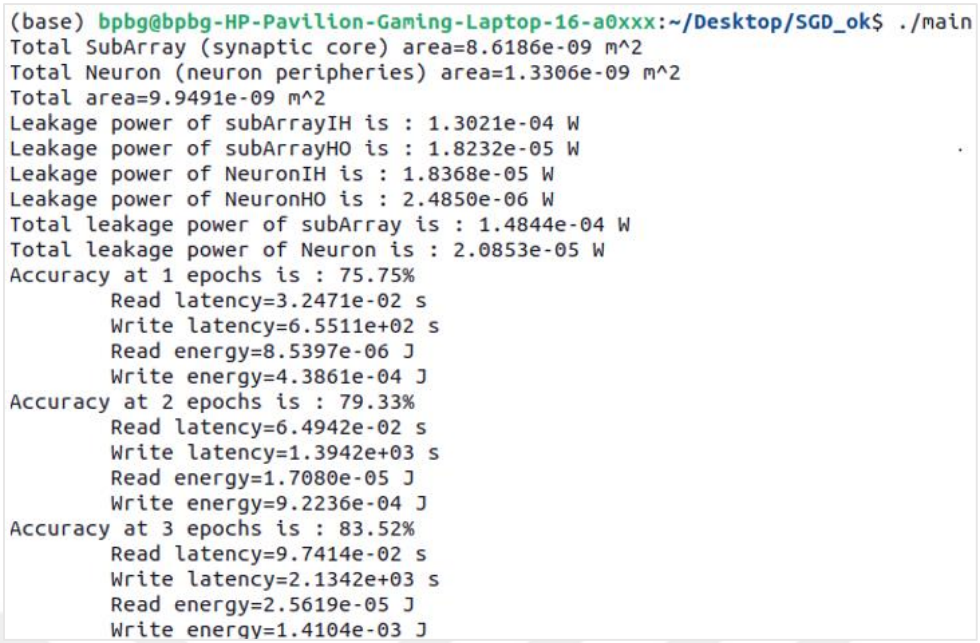
Şekil 31'da uygulamanın Linux işletim sistemi ortamında derlenmesi gösterilmiştir. *"make"* komutuyla başlatılan bu derleme süreci, *GNU Compiler Collection (GCC)* kullanılarak çok sayıda *C++* dosyasının derlenmesini ve nihai olarak tek bir çalıştırılabilir dosya üretimini içermektedir. Bu derleme süreci, yüksek performanslı bir yapay sinir ağı simülasyon ile donanım benzetimi yazılımının hazırlanmasına yöneliktir. Kullanılan optimizasyon seviyesi *(O3)* ve *OpenMP* desteği, performans ihtiyaçlarının karşılanması için seçilmiştir. Bağımsız *object* dosyalarının üretilmesi, modüler yapıyı koruyarak gerekli bileşenlerin her biri ayrı olarak derlenmesini sağlamaktadır.

Donanım tabanlı model, çevrimiçi öğrenme yeteneği ile karakter tanıma yaparken, aynı zamanda çevrimdışı sınıflandırma yapabilme özelliğine sahiptir. Uygulama süreci, bilgisayarla yazılmış ve elle yazılmış rakamları tanımlamak için sinir ağında ileri besleme ve geri yayılma (BP) işlemlerini içermektedir.

```
(base) bpbg@bpbg-HP-Pavilion-Gaming-Laptop-16-a0xxx:~/Desktop/SGD$ make
g++ -c -fopenmp -O3 -std=c++0x -w Array.cpp -o Array.o
g++ -c -fopenmp -O3 -std=c++0x -w Cell.cpp -o Cell.o
g++ -c -fopenmp -O3 -std=c++0x -w formula.cpp -o formula.o
g++ -c -fopenmp -O3 -std=c++0x -w IO.cpp -o IO.o
g++ -c -fopenmp -O3 -std=c++0x -w Mapping.cpp -o Mapping.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim.cpp -o NeuroSim.o
g++ -c -fopenmp -O3 -std=c++0x -w Param.cpp -o Param.o
g++ -c -fopenmp -O3 -std=c++0x -w Test.cpp -o Test.o
g++ -c -fopenmp -O3 -std=c++0x -w Train.cpp -o Train.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/Adder.cpp -o NeuroSim/Adder.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/CurrentSenseAmp.cpp -o NeuroSim/CurrentSenseAmp.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/DecoderDriver.cpp -o NeuroSim/DecoderDriver.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/DFF.cpp -o NeuroSim/DFF.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/formula.cpp -o NeuroSim/formula.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/FunctionUnit.cpp -o NeuroSim/FunctionUnit.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/MultilevelSAEncoder.cpp -o NeuroSim/MultilevelSAEncoder.
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/MultilevelSenseAmp.cpp -o NeuroSim/MultilevelSenseAmp.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/Mux.cpp -o NeuroSim/Mux.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/NewSwitchMatrix.cpp -o NeuroSim/NewSwitchMatrix.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/Precharger.cpp -o NeuroSim/Precharger.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/ReadCircuit.cpp -o NeuroSim/ReadCircuit.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/RowDecoder.cpp -o NeuroSim/RowDecoder.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/SenseAmp.cpp -o NeuroSim/SenseAmp.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/ShiftAdd.cpp -o NeuroSim/ShiftAdd.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/SRAMWriteDriver.cpp -o NeuroSim/SRAMWriteDriver.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/SubArray.cpp -o NeuroSim/SubArray.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/Subtractor.cpp -o NeuroSim/Subtractor.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/SwitchMatrix.cpp -o NeuroSim/SwitchMatrix.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/Technology.cpp -o NeuroSim/Technology.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/VoltageSenseAmp.cpp -o NeuroSim/VoltageSenseAmp.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/WLDecoderOutput.cpp -o NeuroSim/WLDecoderOutput.o
g++ -c -fopenmp -O3 -std=c++0x -w NeuroSim/WLNewDecoderDriver.cpp -o NeuroSim/WLNewDecoderDriver.o
```

Şekil 31. Uygulamanın derlenmesi

Şekil 32'de tez çalışmasında sinir ağı simülasyonunun çalıştırıldığı komut satırı gösterilmektedir. Komut satırında, sinir ağı simülasyonunun bazı donanım özellikleri ve her bir epoch (eğitim döngüsü) için doğruluk oranı, gecikme süreleri ve enerji tüketimi hesaplanarak ekranda gösterilmektedir.



Şekil 32. Uygulamanın çalıştırılması

Şekil 31'de tez çalışmasına ait uygulanın çıktıların üç adımı aşağıda açıklanmıştır.

Alan (Area) Hesaplamaları:

- Total SubArray (synaptic core) area: Sinir ağının ağırlıklarını depolayan çekirdek birimin toplam alanıdır. Burada 8.6186e-09 m² olarak verilmiştir.
- Total Neuron (neuron peripheries) area: Sinir ağındaki nöronların çevresel birimlerinin toplam alanı (örneğin, giriş ve çıkış birimleri). Bu 1.3306e-09 m² olarak hesaplanmıştır.
- Total area: Yukarıdaki iki alanın toplamı olarak 9.9491e-09 m² verilmiş.

Bu alan ölçümleri, donanım mimarisi tasarımında kullanılan bellek ve işlem birimlerinin fiziksel büyüklüklerini ifade etmektedir.

Güç (Power) Hesaplamaları:

- Leakage power of subArrayIH: *"Input-Hidden Layer"* alt dizisindeki sızıntı gücü, 1.3021e-04 W olarak hesaplanmıştır.
- Leakage power of subArrayHO: *"Hidden-Output Layer"* alt dizisindeki sızıntı gücü, 1.8232e-05 W olarak hesaplanmıştır.
- Leakage power of NeuronIH/NeuronHO: Girişten gizli katmana (*IH*) ve gizli katmandan çıkışa (*HO*) olan bağlantılardaki nöronlar için güç tüketimleri verilmiştir.
- Total leakage power: Sistem genelindeki toplam güç sızıntısıdır (1.4844e-04 W subArray için ve 2.0853e-05 W nöronlar için).

Uygulamada elde edilen bu değerler, donanımın çalışma sırasında harcadığı gücü ve sızıntı enerjisini simülasyonla ifade etmektedir.

Doğruluk (Accuracy) ve Performans Verileri:

- Doğruluk (Accuracy), modelin belirli sayıda epoch (iterasyon) sonundaki performansını gösterir:
- Epoch: %75,75 doğruluk
- Epoch: %79,33 doğruluk
- Epoch: %83,52 doğruluk

Performans ölçütleri:

- Read latency: Verilerin okunması sırasında geçen süre (örneğin, ilk epoch (iterasyon) için 3,24712e-02 s).
- Write latency: Verilerin yazılması sırasında geçen süreyi ifade eder.
- Read energy / Write energy: Verilerin okunması ve yazılması sırasında harcanan enerji (joule cinsinden).

Yukarıda Şekil 27'de gösterilen çıktılar açıklanmıştır. Bu çıktılar, sinir ağı tabanlı bir donanım sisteminin enerji verimliliği, alan kullanımı ve performans açısından nasıl çalıştığını detaylı bir şekilde ortaya koymaktadır. Çıktılar, sistem tasarımının fiziksel boyutları (alan), güç tüketimi (leakage power) ve performans ölçütleri (doğruluk, gecikme süreleri, enerji tüketimi) hakkında kapsamlı bilgi sunmaktadır.

Öncelikle, toplam alan hesaplamaları, donanım mimarisinin fiziksel tasarımı ve yerleşim planları açısından oldukça önemlidir. Sinir ağı çekirdeği (synaptic core) ve çevresel nöron birimlerinin ayrı ayrı alan ölçümleri, toplam alanın nasıl dağıldığı gösterilmektedir. Bu alan dağılımı, çip tasarımında hem verimlilik hem de maliyet açısından kritik bir faktördür. SubArray (alt dizi) ve Neuron (nöron) alanlarının birleşimi, tüm sistemin fiziksel büyüklüğünü ortaya koyarken, bu alan ölçümleri, özellikle entegre devrelerde daha az yer kaplayan, daha yüksek yoğunluklu tasarımların geliştirilmesi için önemli bir girdi sağlamaktadır.

Güç tüketimi hesaplamaları, özellikle enerji verimliliği yüksek sistemlerin tasarımında kilit bir role sahiptir. Donanım sisteminin hem *subArray* hem de nöron seviyesindeki sızıntı gücü ölçümleri, sistemin hangi bileşenlerinin daha fazla enerji harcadığını ve bu harcamanın nasıl optimize edilebileceğini anlamaya yardımcı olur. Özellikle, *Input-Hidden Layer (IH)* ve *Hidden-Output Layer (HO)* bağlantılarındaki sızıntı güç değerleri, sinir ağı modelinin çalışması sırasında hangi katmanların daha fazla enerji harcadığını göstermektedir. Toplam güç tüketimi ise donanımın sürekli çalışma sırasında ne kadar enerji gerektirdiğini anlamak için bir referans noktası sağlar.

Model doğruluk ve performans verileri, sinir ağı modelinin belirli *epoch (iterasyon)* sayılarındaki öğrenme yeteneğini ve işlem performansını değerlendirmek için kritik öneme sahiptir. Model, her epoch sonunda doğruluk açısından iyileşme göstermekte ve %75,75'ten başlayarak %83,52'ye kadar yükselmektedir. Bu, modelin eğitim sürecinin etkili olduğunu ve daha fazla epoch ile doğruluk seviyesinin arttığını göstermektedir. Ancak bu artışın, enerji tüketimi ve gecikme süreleri üzerinde bir maliyeti vardır. Okuma ve yazma gecikme süreleri *(read/write latency)* ile enerji tüketimi *(read/write energy)* değerleri, modelin eğitim süreci boyunca nasıl bir yük oluşturduğunu açıkça ortaya koymaktadır. İlk epoch'tan itibaren okuma ve yazma süreleri ile enerji tüketiminde bir artış görülmektedir, bu da daha karmaşık hesaplamaların ve daha büyük veri akışlarının gerçekleştiğine işaret eder.

Tez çalışmasında elde edilen bu çıktılar, donanım ve sinir ağı modelinin birlikte çalışmasından doğan karmaşık bir dengeyi temsil etmektedir. Güç tüketimi ve alan gereksinimlerini azaltmak, doğruluk seviyesini artırırken performans ölçütlerini optimize etmek için kritik önemdedir. Bu bağlamda, veriler, donanım ve yazılım optimizasyonları için başlangıç noktası olarak kullanılabilir. Özellikle enerji verimliliği, doğruluk ve gecikme süresi arasındaki dengenin nasıl iyileştirileceğini anlamak için detaylı bir analizi bu çalışma ile gösterilmiştir. Bu tür bir çalışma hem donanım tasarımcıları hem de sinir ağı algoritması geliştiricileri için yol gösterici bir rehber niteliğindedir.

Uygulamada Elde Edilen Deneysel Sonuçlar

Bu makalede, el yazısı rakam tanıma için memristör tabanlı sinaptik cihazları kullanan bir sinir ağı (NN) modeli kullanılmıştır. Model, 60.000 eğitim örneği ve 0 ile 9 arasında değişen rakamlardan oluşan 10.000 test örneği içeren MNIST veri kümesi üzerinde eğitilmiş ve test işlemine tabi tutulmuştur. MNIST data setinin yanı sıra, bu makalede görüntü sınıflandırması için memristör tabanlı sinaptik cihazlar kullanan bir sinir ağı (NN) modeli kullanılmıştır. Model, 10 sınıfta 50.000 eğitim örneği ve 10.000 test örneği içeren CIFAR-10 veri kümesi üzerinde eğitilmiş ve test edilmiştir. Bu veri setlerine ek olarak modelin kararlılığını pekiştirmek amacıyla Fisher’s Iris veri seti çalışmaya dahil edilmiştir. Devre düzeyindeki performans Neurosim kullanılarak analiz edilmiş ve enerji tüketimi, gecikme süresi ve alan gereksinimleri gibi ölçütler değerlendirilmiştir. Memristör tabanlı makine öğrenimi modeline Adadelta, AdaGrad, Adam, AdaMax, Momentum, Nadam, RMSprop ve SGD dahil olmak üzere çeşitli optimizasyon yöntemleri uygulanmıştır. Her yöntem doğruluğu artırmak için ağırlıkları farklı şekilde günceller. Deneysel sonuçlar Tablo 19’da ve Tablo 20’de gösterilmiştir.

Tablo 19. Optimizasyon modellerinin MNIST veri seti ile karşılaştırmalı performans sonuçları

Optimizasyon Adı	Tüketilen Enerji (J)	Doğruluk Oranı (%)	Eğitim Gecikme Süresi (epoch/s)
AdaDelta	0.0304	89.48	804.58 (s)
AdaGrad	0.0112	79.00	953.69 (s)
Adam	0.2440	79.13	717.19 (s)
AdaMax	0.0115	79.68	851.18 (s)
Momentum	0.1431	88.55	737.09 (s)
Nadam	0.112	81.20	828.19 (s)
SGD	0.0276	89.47	639.34 (s)
RMSprop	0.1603	84.91	740.52 (s)
Geleneksel Bilgisayar	$4.275x10^3$	96.95	140.25 (s)

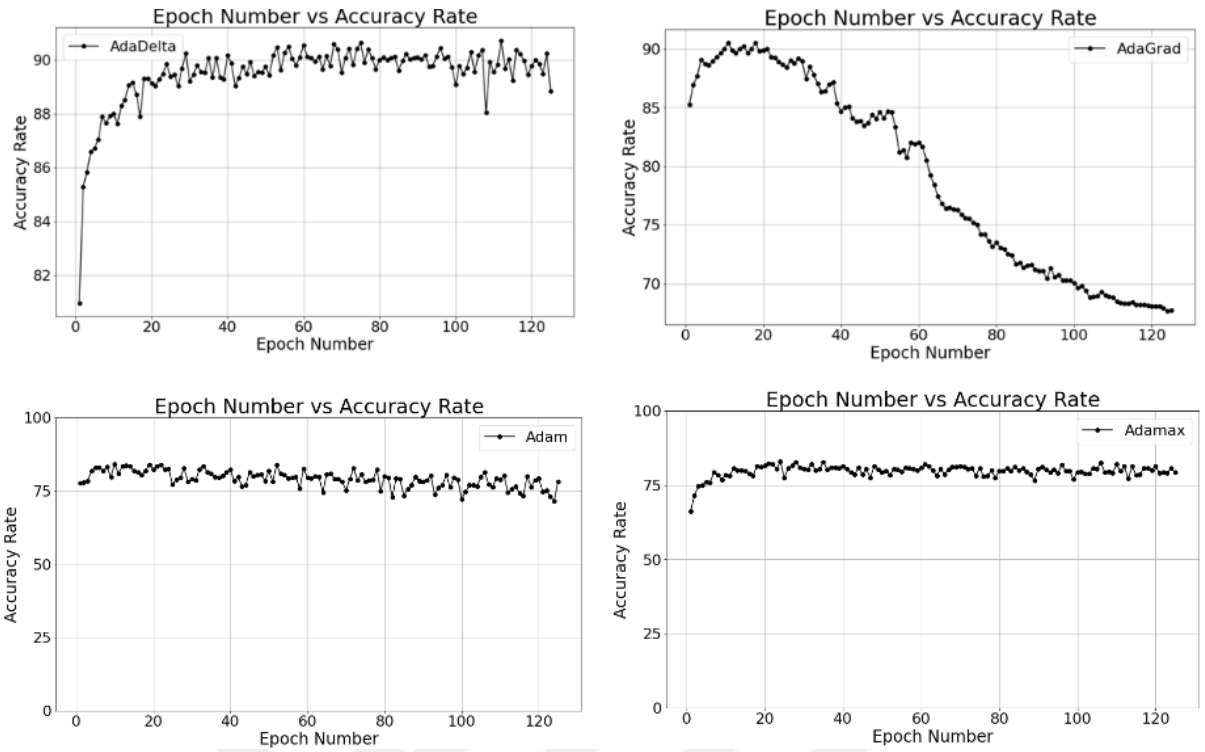
Tablo 20. MNIST, CIFAR ve Fisher's Iris veri setleri ile çeşitli optimizasyon modellerinin karşılaştırmalı doğruluk performans sonuçları

Optimizasyon Adı	Doğruluk Oranı MNIST (%)	Doğruluk Oranı CIAFR-10 (%)	Doğruluk Oranı Fisher's Iris (%)
AdaDelta	89.48	92.51	93.08
AdaGrad	79.00	82.08	81.17
Adam	79.13	83.10	88.23
AdaMax	79.68	81.76	83.71
Momentum	88.55	91.25	92.05
Nadam	81.20	82.45	80.20
SGD	89.47	90.21	90.75
RMSprop	84.91	88.11	87.01

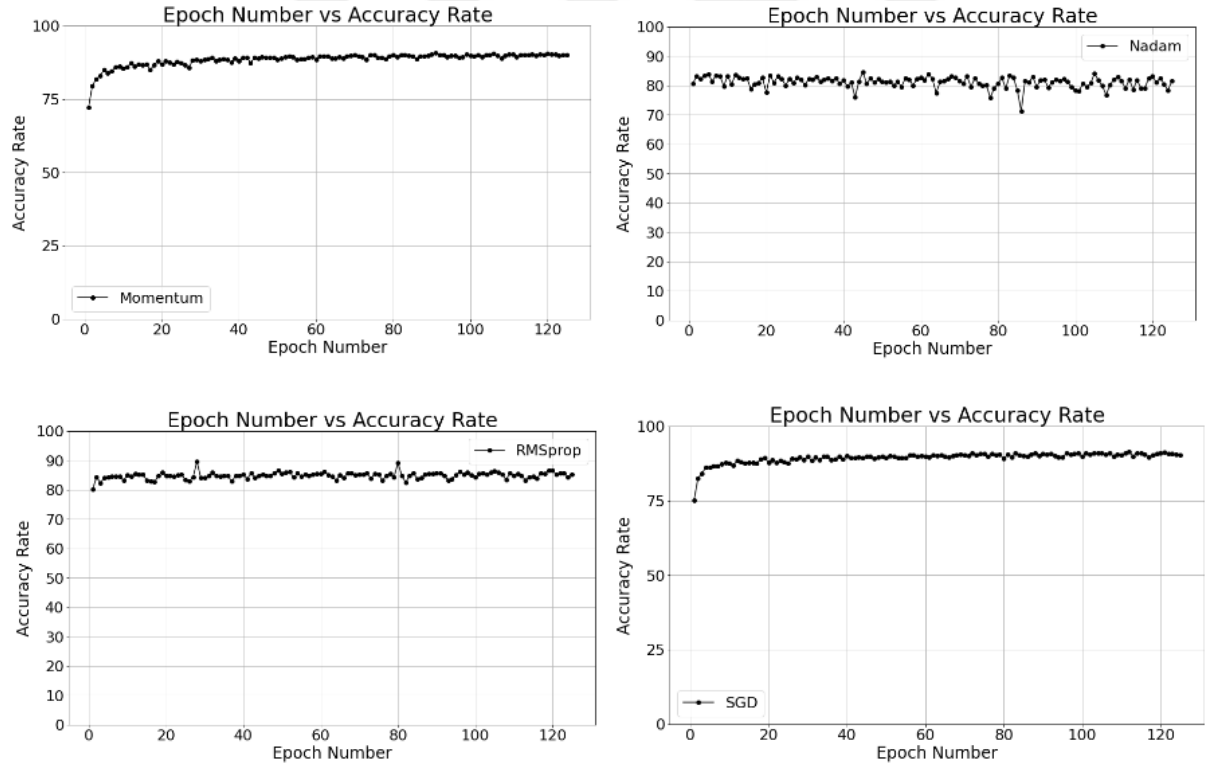
Tablo 21. Sinir ağı mimarilerinin MNIST veri kümesi üzerindeki karşılaştırmalı performans sonuçları.

Mimari	TiO_2 Sinaptik Aygıt NN (Önerilen)	Ag: Si Sinaptik Aygıt NN
Doğruluk (%)	89.48	73.19

Bu tez kapsamında önerilen TiO2 sinaptik tabanlı model Ag: Si sinaptik tabanlı modele göre doğruluk performansı olarak yaklaşık %20 oranında daha iyi sonuç verdiği Tablo 21'de gösterilmiştir.

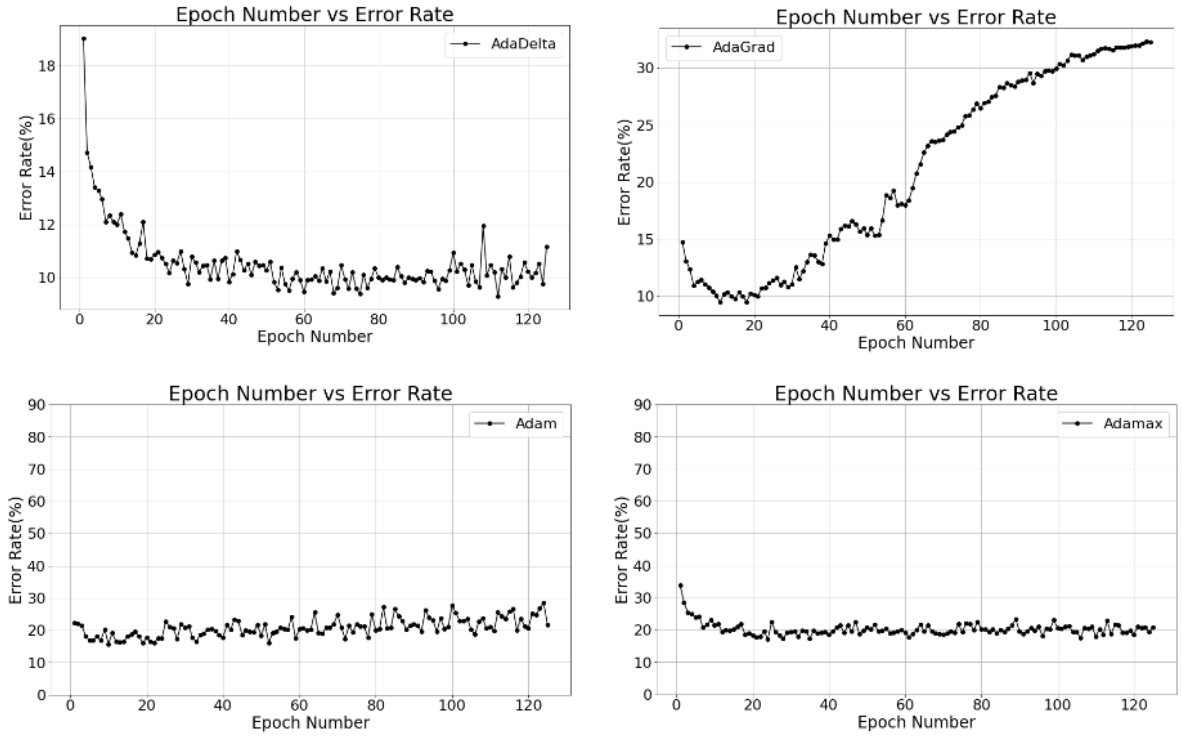


Şekil 33. AdaDelta, AdaGrad, Adam ve Adamax optimizasyon algoritmaları için doğruluk grafikleri

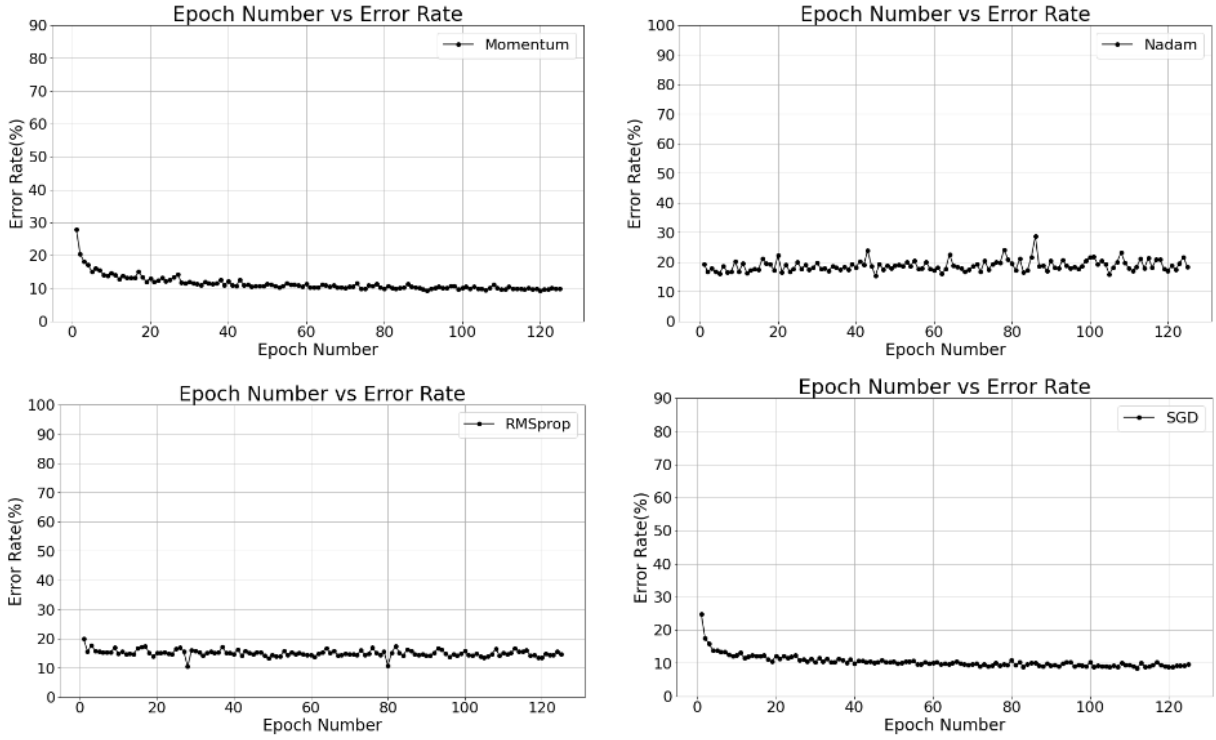


Şekil 34. Momentum, Nadam, RMSProp ve SGD optimizasyon algoritmaları için doğruluk grafikleri

Şekil 33 ve Şekil 34'te tez çalışmasında kullanılan optimizasyon algoritmalarının doğruluk oranları *(Accuracy Rate)* gösterilmiştir.



Şekil 35. AdaDelta, AdaGrad, Adam ve Adamax algoritmaları için hata oranı grafikleri

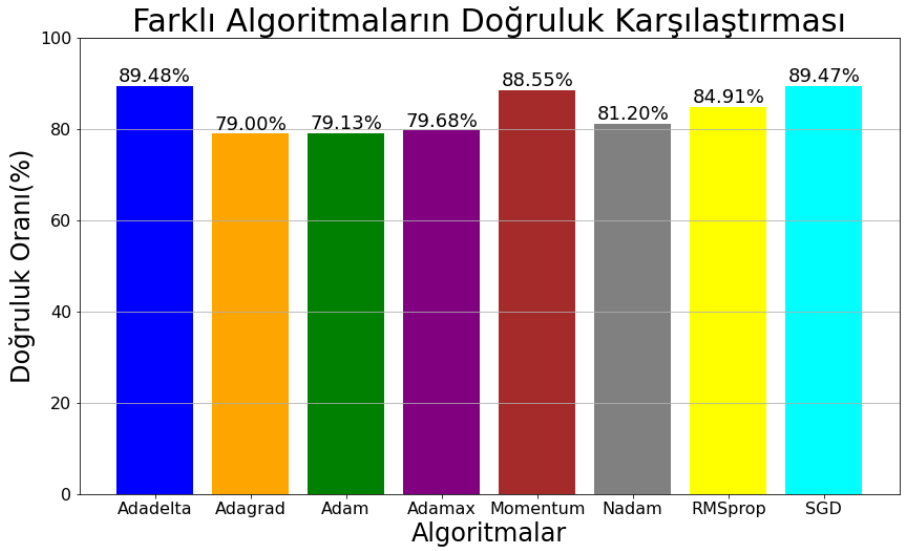


Şekil 36. AdaDelta, AdaGrad, Adam ve Adamax algoritmaları için hata oranı grafikleri

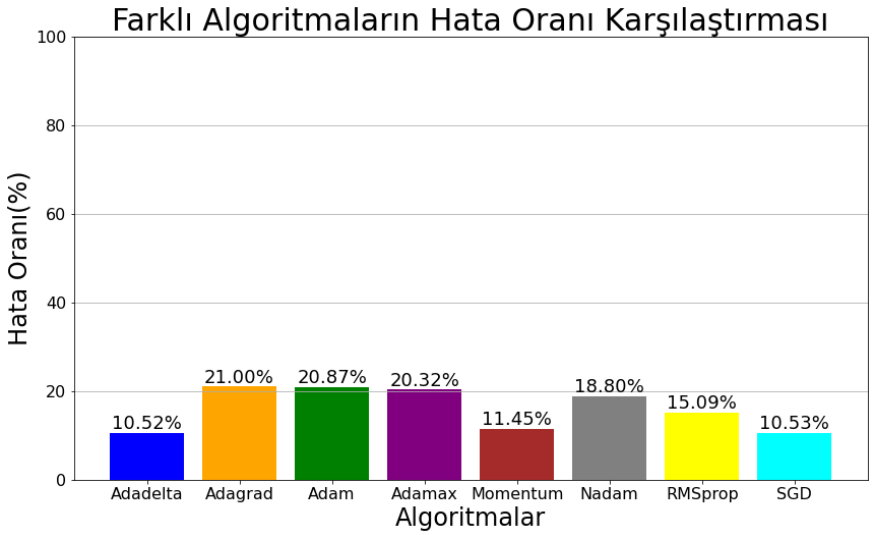
Şekil 35 ve Şekil 36'de tez çalışmasında kullanılan optimizasyon algoritmalarının hata oranları *(Error Rate)* gösterilmiştir. Bu grafikler, farklı optimizasyon yöntemlerinin model performansı üzerindeki kritik etkisini ve yapay sinir ağlarının eğitilmesinde uygun

optimizasyon stratejilerinin seçilmesinin önemini açıkça ortaya koymaktadır. Bu tez çalışması kapsamında uygulamada kullanılan veri setleriyle optimizasyon yöntemlerinin doğruluk üzerinde oluşturduğu etkiler ayrıntılı biçimde incelenmiştir. Bu analizler ışığında, aynı *epoch* sayısında elde edilen doğruluk değerlerindeki önemli farklılıkları görselleştirmektedir. Özellikle, optimizasyon algoritmalarının öğrenme süreçlerini ne ölçüde etkilediğini göstermek açısından bu grafikler oldukça bilgilendiricidir.

Grafiklerde yer alan bulgulara göre, Adadelta algoritması, diğer optimizasyon yöntemlerine kıyasla daha yüksek doğruluk oranları sağlamıştır. Bu sonuç, Adadelta'nın belirli veri kümeleri ve model yapıları için daha etkili bir öğrenme sağladığını göstermektedir. Bununla birlikte, her algoritmanın doğruluk açısından farklı sonuçlar üretmesi, optimizasyon algoritması seçiminin model performansını doğrudan etkileyen kritik bir faktör olduğunu ortaya koymaktadır. Bu grafiklerde sunulan veriler, optimizasyon algoritmaları kullanılarak elde edilen doğruluk ve *epoch* değerlerini içermekte ve bu algoritmaların eğitim aşamasındaki sonuçlarının anlaşılmasına katkı sağlamaktadır. Özellikle, doğruluğu artırmak ve hata oranlarını azaltmak için doğru optimizasyon algoritmasının seçiminin ne denli önemli olduğu vurgulanmaktadır.



Şekil 37. Bu şekil, farklı optimizasyon algoritmalarının doğruluk oranlarını karşılaştırmaktadır. AdaDelta (%89,48), SGD (%89,47) ve Momentum (%88,55) en yüksek doğruluğa sahipken, AdaGrad (%79,00) en düşük doğruluğu göstermektedir.



Şekil 38. Bu şekil, farklı optimizasyon algoritmalarının doğruluk oranlarını karşılaştırmaktadır.

SONUÇLAR VE ÖNERİLER

Sonuç olarak, memristör tabanlı sinaptik cihazlara dayalı sinir ağının performansı, MNIST ve CIFAR veri kümeleri üzerinde çeşitli optimizasyon yöntemleri kullanılarak kapsamlı biçimde incelenmiştir. SGD ve türevleri de dahil olmak üzere farklı optimizasyon algoritmaları test edilmiş ve %90 doğruluk oranı elde edilmiştir. Model, çeşitli optimizasyon yöntemleri arasında sağlamlık ve genelleme kabiliyeti göstermiştir. Bu optimizasyon yöntemleri altındaki bu yüksek performans, veri kümeleri arasındaki uyarlanabilirliği ve etkinliğini vurgulamakta ve çeşitli uygulamalar için potansiyelini göstermektedir. Bu değerlendirme, memristör tabanlı sinaptik cihazların nöromorfik hesaplama ve donanım tabanlı yapay zekâ uygulamaları için çeşitli optimizasyon algoritmalarıyla birleştirilmesinin etkinliğini doğrulamaktadır. Bu bulgular gelecekteki çalışmalara rehberlik ederek daha enerji verimli ve hassas sinir ağı modellerinin oluşturulmasına katkıda bulunmaktadır.

Memristör tabanlı cihazların kullanımı, geleneksel Von Neumann mimarilerinde yaşanan enerji ve zaman verimsizliği sorunlarını önemli ölçüde azaltmaktadır. Bu cihazlar, bellek ve işlem birimlerini birleştirerek veri aktarımını en aza indirirken enerji tüketimini düşürmekte ve hesaplama performansını artırmaktadır. Tez çalışmasında cihazların doğruluk ve enerji tasarrufunu artıran özgün ve modern donanım çözümleri sunduğunu göstermiştir.

Bulgular

Optimizasyon Algoritmaları: Test edilen algoritmalar arasında AdaDelta, CIFAR-10 veri kümesinde %90,51 doğruluk oranıyla en iyi performansı göstermiştir. MNIST veri kümesinde ise SGD (%89,47), AdaDelta (%89,48) ve Momentum (%88,55) algoritmaları yüksek doğruluk oranları sağlamıştır. Bu sonuçlar, optimizasyon yöntemlerinin model performansı üzerinde çok etkili olduğunu göstermektedir.

Cihaz Özellikleri: Memristör tabanlı memristörlerin, CMOS teknolojisiyle uyumlu yapıları sayesinde yüksek yoğunluklu ve enerji verimli donanımlar tasarlanmasına olanak tanıdığı gözlemlenmiştir. Bu cihazların biyolojik sinapsları taklit ederek direnç seviyelerini dinamik biçimde ayarlayabilmesi, nöromorfik uygulamalar için büyük bir avantaj sunmaktadır.

Enerji Verimliliği: Bu tez çalışmasıyla memristör tabanlı mimariler, geleneksel sistemlere kıyasla enerji tüketimini önemli ölçüde azaltmıştır. Enerji tüketiminin büyük kısmı statik güç kullanımıyla ilişkilendirilmiş, bu da memristörlerin dinamik güç tüketimine kıyasla avantajlı olduğunu göstermiştir.

Doğruluk-Enerji Dengesi: Bu çalışmada yüksek doğruluk oranları elde edilmesine rağmen, bu doğruluğun enerji tüketimi üzerindeki etkileri kapsamlı şekilde incelenmiştir. Özellikle AdaDelta algoritması hem doğruluk hem de enerji verimliliği açısından üstün performans sergileyerek algoritma seçimlerinin nöromorfik donanım tasarımı üzerindeki önemini vurgulamaktadır.

Genelleme Kabiliyeti: Bu tez çalışmasında ele alınan model, farklı veri kümelerinde sağlam bir genelleme kapasitesi göstermiştir. Bu durum, memristör tabanlı cihazların geniş bir makine öğrenimi görev yelpazesinde etkili biçimde kullanılabileceğini göstermektedir.

Öneriler

Bu çalışma, memristör tabanlı memristör cihazların nöromorfik hesaplama ve donanım tabanlı yapay zekâ uygulamalarındaki potansiyelini açık biçimde ortaya koymaktadır. Elde edilen bulgular, aşağıdaki alanlarda gelecekteki araştırmalara yol gösterici olabilir:

Ölçeklenebilirlik: Memristör tabanlı cihazların daha büyük sinir ağları ve daha karmaşık veri kümeleri için uygulanabilirliği araştırılmalıdır. Özellikle ağırlık hassasiyeti ve kuantizasyon sorunları üzerine çalışmalar yapılabilir.

Enerji Optimizasyonu: Enerji sarfiyatını daha da azaltmak amacıyla yeni yöntemlerin geliştirilmesi, nöromorfik sistemlerin enerji tasarrufunu artırabilir.

Algoritma-Donanım Ortak Tasarımı: Optimizasyon algoritmalarının donanım özelliklerine uygun biçimde geliştirilmesi, performansı daha da iyileştirebilir. Örneğin, AdaDelta ve SGD'nin avantajlarını birleştiren hibrit yaklaşımlar daha başarılı sonuçlar sağlayabilir.

Gerçek Dünya Uygulamaları: Memristör tabanlı cihazların kenar yapay zekâ, otonom sistemler ve beyin ilhamlı hesaplama gibi alanlarda uygulanabilirliği genişletilmelidir.

Bu tez çalışması, memristör tabanlı memristör cihazların nöromorfik hesaplama için uygulanabilirliğini doğrulamış ve enerji verimliliği ile doğruluk arasında güçlü bir denge sağladığını göstermiştir. Çalışmanın sonuçları, donanım hızlandırmalı yapay zekâ sistemleri için sürdürülebilir ve ölçeklenebilir çözümler geliştirmek adına önemli bir temel sunmaktadır. Bu bulgular, daha enerji verimli, doğru ve genel kullanıma uygun sinir ağı modellerinin tasarlanmasına katkıda bulunacaktır.

KAYNAKLAR

Adhikari, S. P., Sah, M. P., Kim, H., & Chua, L. O. (2013). Three Fingerprints of Memristor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *60*(11), 3008–3021. https://doi.org/10.1109/TCSI.2013.2256171

Al-Shedivat, M., Naous, R., Cauwenberghs, G., & Salama, K. N. (2015). Memristors Empower Spiking Neurons With Stochasticity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *5*(2), 242–253. https://doi.org/10.1109/JETCAS.2015.2435512

Alibart, F., Pleutin, S., Bichler, O., Gamrat, C., Serrano-Gotarredona, T., Linares-Barranco, B., & Vuillaume, D. (2012). A Memristive Nanoparticle/Organic Hybrid Synapstor for Neuroinspired Computing. *Advanced Functional Materials*, *22*(3), 609–616. https://doi.org/10.1002/adfm.201101935

Ambrogio, S., Narayanan, P., Okazaki, A., Fasoli, A., Mackin, C., Hosokawa, K., Nomura, A., Yasuda, T., Chen, A., Friz, A., Ishii, M., Luquin, J., Kohda, Y., Saulnier, N., Brew, K., Choi, S., Ok, I., Philip, T., Chan, V., … Burr, G. W. (2023). An analog-AI chip for energy-efficient speech recognition and transcription. *Nature*, *620*(7975), 768–775. https://doi.org/10.1038/s41586-023-06337-5

Amirsoleimani, A., Alibart, F., Yon, V., Xu, J., Pazhouhandeh, M. R., Ecoffey, S., Beilliard, Y., Genov, R., & Drouin, D. (2020). In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. *Advanced Intelligent Systems*, *2*(11), 2000115. https://doi.org/10.1002/aisy.202000115

Anand, V., Gupta, S., Altameem, A., Nayak, S. R., Poonia, R. C., & Saudagar, A. K. J. (2022). An Enhanced Transfer Learning Based Classification for Diagnosis of Skin Cancer. *Diagnostics*, *12*(7), 1628. https://doi.org/10.3390/diagnostics12071628

Anti-Hebbian and Hebbian (AHaH) Plasticity. (2017). Knowm. https://knowm.org/ahah-computing/

Ascoli, A., Tetzlaff, R., Chua, L. O., Strachan, J. P., & Williams, R. S. (2016). History Erase Effect in a Non-Volatile Memristor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *63*(3), 389–400. https://doi.org/10.1109/TCSI.2016.2525043

Babacan, Y., Kaçar, F., & Gürkan, K. (2016). A spiking and bursting neuron circuit based on memristor. *Neurocomputing*, *203*, 86–91. https://doi.org/10.1016/j.neucom.2016.03.060

Bao, B., Hu, A., Bao, H., Xu, Q., Chen, M., & Wu, H. (2018). Three-Dimensional Memristive Hindmarsh–Rose Neuron Model with Hidden Coexisting Asymmetric Behaviors. *Complexity*, *2018*, 1–11. https://doi.org/10.1155/2018/3872573

Bear, M. F., & Malenka, R. C. (1994). Synaptic plasticity: LTP and LTD. *Current Opinion in Neurobiology*, *4*(3), 389–399. https://doi.org/10.1016/0959-4388(94)90101-5

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bian, K., & Priyadarshi, R. (2024a). Machine Learning Optimization Techniques: A Survey, Classification, Challenges, and Future Research Issues. *Archives of Computational*

Methods in Engineering. https://doi.org/10.1007/s11831-024-10110-w

Bian, K., & Priyadarshi, R. (2024b). Machine Learning Optimization Techniques: A Survey, Classification, Challenges, and Future Research Issues. *Archives of Computational Methods in Engineering*. https://doi.org/10.1007/s11831-024-10110-w

Binelli, E., Broggi, A., Fascioli, A., Ghidoni, S., Grisleri, P., Graf, T., & Meinecke, M. (2005). A modular tracking system for far infrared pedestrian recognition. *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, 759–764. https://doi.org/10.1109/IVS.2005.1505196

Biolek, D., & Biolková, V. (2009). *SPICE Model of Memristor with Nonlinear Dopant Drift*. *1*, 210–214. http://hdl.handle.net/11012/57100

Bliss, T. V. P., & Lømo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of Physiology*, *232*(2), 331–356. https://doi.org/10.1113/jphysiol.1973.sp010273

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., & Muller, U. (2017). *Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car*. http://arxiv.org/abs/1704.07911

Borghetti, J., Li, Z., Straznicky, J., Li, X., Ohlberg, D. A. A., Wu, W., Stewart, D. R., & Williams, R. S. (2009). A hybrid nanomemristor/transistor logic circuit capable of self-programming. *Proceedings of the National Academy of Sciences*, *106*(6), 1699–1703. https://doi.org/10.1073/pnas.0806642106

Bower, J. M., & Beeman, D. (1995). *The Book of GENESIS: Exploring Realistic Neural Models with the General Neural Simulation System*. Springer.

Burr, G. W., Shelby, R. M., Sebastian, A., Kim, S., Kim, S., Sidler, S., Virwani, K., Ishii, M., Narayanan, P., Fumarola, A., Sanches, L. L., Boybat, I., Le Gallo, M., Moon, K., Woo, J., Hwang, H., & Leblebici, Y. (2017). Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, *2*(1), 89–124. https://doi.org/10.1080/23746149.2016.1259585

Cai, F., Correll, J. M., Lee, S. H., Lim, Y., Bothra, V., Zhang, Z., Flynn, M. P., & Lu, W. D. (2019). A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nature Electronics*, *2*(7), 290–299. https://doi.org/10.1038/s41928-019-0270-x

Cai, Y., Wu, J., & Zhang, C. (2019). Classification of Trash Types in Cotton Based on Deep Learning. *2019 Chinese Control Conference (CCC)*, 8783–8788. https://doi.org/10.23919/ChiCC.2019.8865475

Cai, Z., Zhu, X., Gergondet, P., Chen, X., & Yu, Z. (2023). A Friction-Driven Strategy for Agile Steering Wheel Manipulation by Humanoid Robots. *Cyborg and Bionic Systems*, *4*. https://doi.org/10.34133/cbsystems.0064

Cao, Z., Sun, B., Zhou, G., Mao, S., Zhu, S., Zhang, J., Ke, C., Zhao, Y., & Shao, J. (2023). Memristor-based neural networks: a bridge from device to artificial intelligence. *Nanoscale Horizons*, *8*(6), 716–745. https://doi.org/10.1039/D2NH00536K

Caporale, N., & Dan, Y. (2008). Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, *31*(1), 25–46. https://doi.org/10.1146/annurev.neuro.31.060407.125639

Capra, M., Bussolino, B., Marchisio, A., Masera, G., Martina, M., & Shafique, M. (2020). Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead. *IEEE Access*, *8*, 225134–225180. https://doi.org/10.1109/ACCESS.2020.3039858

Caravelli, F., & Carbajal, J. (2018). Memristors for the Curious Outsiders. *Technologies*, *6*(4), 118. https://doi.org/10.3390/technologies6040118

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible Models for HealthCare. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1721–1730. https://doi.org/10.1145/2783258.2788613

Chen, C.-Y., Shih, H.-C., Wu, C.-W., Lin, C.-H., Chiu, P.-F., Sheu, S.-S., & Chen, F. T. (2015). RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme. *IEEE Transactions on Computers*, *64*(1), 180–190. https://doi.org/10.1109/TC.2014.12

Chen, C. Y., & Chakrabarty, K. (2021). Pruning of Deep Neural Networks for Fault-Tolerant Memristor-based Accelerators. *Proceedings - Design Automation Conference*, *2021-Decem*, 889–894. https://doi.org/10.1109/DAC18074.2021.9586269

Chen, J., Li, J., Li, Y., & Miao, X. (2021). Multiply accumulate operations in memristor crossbar arrays for analog computing. *Journal of Semiconductors*, *42*(1), 013104. https://doi.org/10.1088/1674-4926/42/1/013104

Chen, P.-Y., Peng, X., & Yu, S. (2018). NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *37*(12), 3067–3080. https://doi.org/10.1109/TCAD.2018.2789723

Chen, S., Zhang, T., Tappertzhofen, S., Yang, Y., & Valov, I. (2023). Electrochemical-Memristor-Based Artificial Neurons and Synapses—Fundamentals, Applications, and Challenges. *Advanced Materials*, *35*(37). https://doi.org/10.1002/adma.202301924

Cheng, G., & An, X. (2021a). A Brief Overview of Deep Learning and Memristor. *Journal of Physics: Conference Series*, *1894*(1), 012086. https://doi.org/10.1088/1742-6596/1894/1/012086

Cheng, G., & An, X. (2021b). A Brief Overview of Deep Learning and Memristor. *Journal of Physics: Conference Series*, *1894*(1). https://doi.org/10.1088/1742-6596/1894/1/012086

Chiu, Y.-C., Khwa, W.-S., Yang, C.-S., Teng, S.-H., Huang, H.-Y., Chang, F.-C., Wu, Y., Chien, Y.-A., Hsieh, F.-L., Li, C.-Y., Lin, G.-Y., Chen, P.-J., Pan, T.-H., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Ho, M.-S., Lo, C.-P., … Chang, M.-F. (2023). A CMOS-integrated spintronic compute-in-memory macro for secure AI edge devices. *Nature Electronics*, *6*(7), 534–543. https://doi.org/10.1038/s41928-023-00994-0

Choe, J., & Shim, H. (2019). *Attention-based Dropout Layer for Weakly Supervised Object Localization*. https://doi.org/https://doi.org/10.48550/arXiv.1908.10028

Choi, C., Kim, H., Kang, J.-H., Song, M.-K., Yeon, H., Chang, C. S., Suh, J. M., Shin, J., Lu, K., Park, B.-I., Kim, Y., Lee, H. E., Lee, D., Lee, J., Jang, I., Pang, S., Ryu, K., Bae, S.-H., Nie, Y., … Kim, J. (2022). Reconfigurable heterogeneous integration using stackable chips with embedded artificial intelligence. *Nature Electronics*, *5*(6), 386–393.

https://doi.org/10.1038/s41928-022-00778-y

Chua, L. (1971a). Memristor-The missing circuit element. *IEEE Transactions on Circuit Theory*, *18*(5), 507–519. https://doi.org/10.1109/TCT.1971.1083337

Chua, L. (1971b). Memristor-The missing circuit element. *IEEE Transactions on Circuit Theory*, *18*(5), 507–519. https://doi.org/10.1109/TCT.1971.1083337

Chua, L. O., & Sung Mo Kang. (1976). Memristive devices and systems. *Proceedings of the IEEE*, *64*(2), 209–223. https://doi.org/10.1109/PROC.1976.10092

Corinto, F., Ascoli, A., & Sung-Mo Kang. (2013). Memristor-based neural circuits. *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, 417–420. https://doi.org/10.1109/ISCAS.2013.6571869

Corinto, F., Civalleri, P. P., & Chua, L. O. (2015). A Theoretical Approach to Memristor Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *5*(2), 123–132. https://doi.org/10.1109/JETCAS.2015.2426494

Cruz-Albrecht, J. M., Yung, M. W., & Srinivasa, N. (2012). Energy-efficient neuron, synapse and STDP integrated circuits. *IEEE Transactions on Biomedical Circuits and Systems*, *6*(3), 246–256. https://doi.org/10.1109/TBCAS.2011.2174152

Cumming, D. R. S., Furber, S. B., & Paul, D. J. (2014). Beyond Moore's law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *372*(2012), 20130376. https://doi.org/10.1098/rsta.2013.0376

Dan, Y., & Poo, M.-M. (2006). Spike Timing-Dependent Plasticity: From Synapse to Perception. *Physiological Reviews*, *86*(3), 1033–1048. https://doi.org/10.1152/physrev.00030.2005

Daoudal, G., & Debanne, D. (2003). Long-Term Plasticity of Intrinsic Excitability: Learning Rules and Mechanisms. *Learning & Memory*, *10*(6), 456–465. https://doi.org/10.1101/lm.64103

del Valle, J., Ramírez, J. G., Rozenberg, M. J., & Schuller, I. K. (2018). Challenges in materials and devices for resistive-switching-based neuromorphic computing. *Journal of Applied Physics*, *124*(21). https://doi.org/10.1063/1.5047800

Deng, L. (2014). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197–387. https://doi.org/10.1561/2000000039

Deng, L., Wang, D., Zhang, Z., Tang, P., Li, G., & Pei, J. (2016). Energy consumption analysis for various memristive networks under different learning strategies. *Physics Letters A*, *380*(7–8), 903–909. https://doi.org/10.1016/j.physleta.2015.12.024

Dennard, R. H., Gaensslen, F. H., Yu, H.-N., Rideout, V. L., Bassous, E., & LeBlanc, A. R. (1974). Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, *9*(5), 256–268. https://doi.org/10.1109/JSSC.1974.1050511

Do, K., Lee, D., Ko, D.-H., Sohn, H., & Cho, M.-H. (2010). TEM Study on Volume Changes and Void Formation in Ge[sub 2]Sb[sub 2]Te[sub 5] Films, with Repeated Phase Changes. *Electrochemical and Solid-State Letters*, *13*(8), H284. https://doi.org/10.1149/1.3439647

Drakopoulos, F., Baby, D., & Verhulst, S. (2021). A convolutional neural-network framework

for modelling auditory sensory cells and synapses. *Communications Biology*, *4*(1), 827. https://doi.org/10.1038/s42003-021-02341-5

Dretchen, K. L., Standaert, F. G., Skirboll, L. R., & Morgenroth, V. H. (1976). Evidence for a prejunctional role of cyclic nucleotides in neuromuscular transmission. *Nature*, *264*(5581), 79–81. https://doi.org/10.1038/264079a0

Du, Z., Ben-Dayan Rubin, D. D., Chen, Y., He, L., Chen, T., Zhang, L., Wu, C., & Temam, O. (2015). Neuromorphic accelerators. *Proceedings of the 48th International Symposium on Microarchitecture*, 494–507. https://doi.org/10.1145/2830772.2830789

Eryilmaz, S. B., Kuzum, D., Jeyasingh, R. G. D., Kim, S., BrightSky, M., Lam, C., & Wong, H.-S. P. (2013). Experimental demonstration of array-level learning with phase change synaptic devices. *2013 IEEE International Electron Devices Meeting*, 25.5.1-25.5.4. https://doi.org/10.1109/IEDM.2013.6724691

Eryilmaz, S. B., Kuzum, D., Jeyasingh, R., Kim, S., BrightSky, M., Lam, C., & Wong, H.-S. P. (2014a). Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Frontiers in Neuroscience*, *8*. https://doi.org/10.3389/fnins.2014.00205

Eryilmaz, S. B., Kuzum, D., Jeyasingh, R., Kim, S., BrightSky, M., Lam, C., & Wong, H.-S. P. (2014b). Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Frontiers in Neuroscience*, *8*(8 JUL), 1–11. https://doi.org/10.3389/fnins.2014.00205

Eshraghian, J. K., Wang, X., & Lu, W. D. (2022a). Memristor-Based Binarized Spiking Neural Networks: Challenges and applications. *IEEE Nanotechnology Magazine*, *16*(2), 14–23. https://doi.org/10.1109/MNANO.2022.3141443

Eshraghian, J. K., Wang, X., & Lu, W. D. (2022b). Memristor-Based Binarized Spiking Neural Networks: Challenges and applications. *IEEE Nanotechnology Magazine*, *16*(2), 14–23. https://doi.org/10.1109/MNANO.2022.3141443

Feali, M. S. (2021). Using volatile/non-volatile memristor for emulating the short-and long-term adaptation behavior of the biological neurons. *Neurocomputing*, *465*, 157–166. https://doi.org/10.1016/j.neucom.2021.08.132

Feldman, D. E. (2012). The Spike-Timing Dependence of Plasticity. *Neuron*, *75*(4), 556–571. https://doi.org/10.1016/j.neuron.2012.08.001

Feng, J., Shen, L., Chen, Z., Wang, Y., & Li, H. (2020). A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic. *IEEE Access*, *8*, 125786–125796. https://doi.org/10.1109/ACCESS.2020.3008081

Gale, E. (2014). TiO 2 -based memristors and ReRAM: materials, mechanisms and models (a review). *Semiconductor Science and Technology*, *29*(10), 104004. https://doi.org/10.1088/0268-1242/29/10/104004

Gao, B., Kang, J., Zhou, Z., Chen, Z., Huang, P., Liu, L., & Liu, X. (2016). Metal oxide resistive random access memory based synaptic devices for brain-inspired computing. *Japanese Journal of Applied Physics*, *55*(4S), 04EA06. https://doi.org/10.7567/JJAP.55.04EA06

Gao, L., Wang, I.-T., Chen, P.-Y., Vrudhula, S., Seo, J., Cao, Y., Hou, T.-H., & Yu, S. (2015). Fully parallel write/read in resistive synaptic array for accelerating on-chip

learning. *Nanotechnology*, *26*(45), 455204. https://doi.org/10.1088/0957-4484/26/45/455204

García-Martín, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, *134*, 75–88. https://doi.org/10.1016/j.jpdc.2019.07.007

Gharpinde, R., Thangkhiew, P. L., Datta, K., & Sengupta, I. (2018). A Scalable In-Memory Logic Synthesis Approach Using Memristor Crossbar. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *26*(2), 355–366. https://doi.org/10.1109/TVLSI.2017.2763171

Gkoupidenis, P., Schaefer, N., Strakosas, X., Fairfield, J. A., & Malliaras, G. G. (2015). Synaptic plasticity functions in an organic electrochemical transistor. *Applied Physics Letters*, *107*(26), 263302. https://doi.org/10.1063/1.4938553

Golonzka, O., Arslan, U., Bai, P., Bohr, M., Baykan, O., Chang, Y., Chaudhari, A., Chen, A., Clarke, J., Connor, C., Das, N., English, C., Ghani, T., Hamzaoglu, F., Hentges, P., Jain, P., Jezewski, C., Karpov, I., Kothari, H., … Fischer, K. (2019). Non-Volatile RRAM Embedded into 22FFL FinFET Technology. *2019 Symposium on VLSI Technology*, T230–T231. https://doi.org/10.23919/VLSIT.2019.8776570

Gong, Q., Kang, W., & Fahroo, F. (2023). Approximation of compositional functions with ReLU neural networks. *Systems & Control Letters*, *175*, 105508. https://doi.org/10.1016/j.sysconle.2023.105508

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *deep learning English version* (Vol. 26).

Greenberg-Toledo, T., Mazor, R., Haj-Ali, A., & Kvatinsky, S. (2019). Supporting the Momentum Training Algorithm Using a Memristor-Based Synapse. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *66*(4), 1571–1583. https://doi.org/10.1109/TCSI.2018.2888538

Gul, F. (2019). Circuit Implementation of Nano-Scale TiO 2 Memristor Using Only Metal-Oxide-Semiconductor Transistors. *IEEE Electron Device Letters*, *40*(4), 643–646. https://doi.org/10.1109/LED.2019.2899889

Gul, F. (2020). Nano-scale single layer TiO2-based artificial synaptic device. *Applied Nanoscience*, *10*(2), 611–616. https://doi.org/10.1007/s13204-019-01179-y

Gupta, V., Mishra, V. K., Singhal, P., & Kumar, A. (2022). An Overview of Supervised Machine Learning Algorithm. *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, 87–92. https://doi.org/10.1109/SMART55829.2022.10047618

Han, J., Yun, S., Lee, S., Yu, J., & Choi, Y. (2022). A Review of Artificial Spiking Neuron Devices for Neural Processing and Sensing. *Advanced Functional Materials*, *32*(33). https://doi.org/10.1002/adfm.202204102

Hansen, K., Baehrens, D., Schroeter, T., Rupp, M., & Müller, K.-R. (2011). Visual Interpretation of Kernel-Based Prediction Models. *Molecular Informatics*, *30*(9), 817–826. https://doi.org/10.1002/minf.201100059

Haoxiang, W., & S, S. (2021). Overview of Configuring Adaptive Activation Functions for Deep Neural Networks - A Comparative Study. *Journal of Ubiquitous Computing and Communication Technologies*, *3*(1), 10–22. https://doi.org/10.36548/jucct.2021.1.002

Hasler, J., & Marr, B. (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience*, *7*. https://doi.org/10.3389/fnins.2013.00118

Hinduja, D., Dheebhika, R., & Jacob, T. P. (2019). Enhanced Character Recognition using Deep Neural Network-A Survey. *2019 International Conference on Communication and Signal Processing (ICCSP)*, 0438–0440. https://doi.org/10.1109/ICCSP.2019.8698008

Ho, Y., Huang, G. M., & Li, P. (2011). Dynamical Properties and Design Analysis for Nonvolatile Memristor Memories. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *58*(4), 724–736. https://doi.org/10.1109/TCSI.2010.2078710

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoon Chung, Sung Joo Lee, & Jeon Gue Park. (2016). Deep neural network using trainable activation functions. *2016 International Joint Conference on Neural Networks (IJCNN)*, *2016-Octob*(l), 348–352. https://doi.org/10.1109/IJCNN.2016.7727219

Hu, M., Chen, Y., Yang, J. J., Wang, Y., & Li, H. H. (2017). A Compact Memristor-Based Dynamic Synapse for Spiking Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *36*(8), 1353–1366. https://doi.org/10.1109/TCAD.2016.2618866

Hu, M., Strachan, J. P., Li, Z., Grafals, E. M., Davila, N., Graves, C., Lam, S., Ge, N., Yang, J. J., & Williams, R. S. (2016). Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. *Proceedings - Design Automation Conference*, *05-09-June*. https://doi.org/10.1145/2897937.2898010

Huang, Y., Ravichandran, V., Zhao, W., & Xia, Q. (2023). Towards Energy-Efficient Computing Hardware Based on Memristive Nanodevices. *IEEE Nanotechnology Magazine*, *17*(5), 30–38. https://doi.org/10.1109/MNANO.2023.3297106

Huh, W., Lee, D., & Lee, C. (2020). Memristors Based on 2D Materials as an Artificial Synapse for Neuromorphic Electronics. *Advanced Materials*, *32*(51), 2002092. https://doi.org/10.1002/adma.202002092

Hung, J.-M., Xue, C.-X., Kao, H.-Y., Huang, Y.-H., Chang, F.-C., Huang, S.-P., Liu, T.-W., Jhang, C.-J., Su, C.-I., Khwa, W.-S., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Ho, M.-S., Chou, C.-C., Chih, Y.-D., Chang, T.-Y. J., & Chang, M.-F. (2021). A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices. *Nature Electronics*, *4*(12), 921–930. https://doi.org/10.1038/s41928-021-00676-9

Ielmini, D., & Wong, H.-S. P. (2018). In-memory computing with resistive switching devices. *Nature Electronics*, *1*(6), 333–343. https://doi.org/10.1038/s41928-018-0092-2

Illarionov, G. A., Morozova, S. M., Chrishtop, V. V., Einarsrud, M.-A., & Morozov, M. I. (2020). Memristive TiO2: Synthesis, Technologies, and Applications. *Frontiers in Chemistry*, *8*. https://doi.org/10.3389/fchem.2020.00724

Ilyas, N., Li, D., Li, C., Jiang, X., Jiang, Y., & Li, W. (2020). Analog Switching and Artificial Synaptic Behavior of Ag/SiOx:Ag/TiOx/p++-Si Memristor Device. *Nanoscale Research Letters*, *15*(1), 0–10. https://doi.org/10.1186/s11671-020-3249-7

Indiveri, G. (2003). A low-power adaptive integrate-and-fire neuron circuit. *Proceedings - IEEE International Symposium on Circuits and Systems*, *4*.

https://doi.org/10.1109/iscas.2003.1206342

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Schaik, A. van, Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic Silicon Neuron Circuits. *Frontiers in Neuroscience*, *5*. https://doi.org/10.3389/fnins.2011.00073

Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, *404*, 109136. https://doi.org/10.1016/j.jcp.2019.109136

Jeong, D. S., Kim, K. M., Kim, S., Choi, B. J., & Hwang, C. S. (2016). Memristors for Energy-Efficient New Computing Paradigms. *Advanced Electronic Materials*, *2*(9). https://doi.org/10.1002/aelm.201600090

Jhaveri, R. H., Revathi, A., Ramana, K., Raut, R., & Dhanaraj, R. K. (2022). A Review on Machine Learning Strategies for Real-World Engineering Applications. *Mobile Information Systems*, *2022*, 1–26. https://doi.org/10.1155/2022/1833507

Jin, S., Pei, S., & Wang, Y. (2020). A variation tolerant scheme for memristor crossbar based neural network designs via two-phase weight mapping and memristor programming. *Future Generation Computer Systems*, *106*, 270–276. https://doi.org/10.1016/j.future.2020.01.021

Jin, Y.-H., Cai, L., Cheng, Z.-S., Cheng, H., Deng, T., Fan, Y.-P., Fang, C., Huang, D., Huang, L.-Q., Huang, Q., Han, Y., Hu, B., Hu, F., Li, B.-H., Li, Y.-R., Liang, K., Lin, L.-K., Luo, L.-S., Ma, J., … Wang, X.-H. (2020). A rapid advice guideline for the diagnosis and treatment of 2019 novel coronavirus (2019-nCoV) infected pneumonia (standard version). *Military Medical Research*, *7*(1), 4. https://doi.org/10.1186/s40779-020-0233-6

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., & Lu, W. (2010). Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Letters*, *10*(4), 1297–1301. https://doi.org/10.1021/nl904092h

Johnsen, G. K. (2012). An introduction to the memristor – a valuable circuit element in bioelectricity and bioimpedance. *Journal of Electrical Bioimpedance*, *3*(1), 20–28. https://doi.org/10.5617/jeb.305

Joksas, D., Freitas, P., Chai, Z., Ng, W. H., Buckwell, M., Li, C., Zhang, W. D., Xia, Q., Kenyon, A. J., & Mehonic, A. (2020). Committee machines—a universal method to deal with non-idealities in memristor-based neural networks. *Nature Communications*, *11*(1), 4273. https://doi.org/10.1038/s41467-020-18098-0

Josh, P., & Gibson, A. (2017). *Deep Learning A Practitioner's Approach* (p. 800). O'Reilly Media.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., … Yoon, D. H. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12. https://doi.org/10.1145/3079856.3080246

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., & Hudspeth, A. J. (2012).

Principles of Neural Science (5th ed.). McGraw-Hill Education.

Kaneko, Y., Nishitani, Y., & Ueda, M. (2014). Ferroelectric Artificial Synapses for Recognition of a Multishaded Image. *IEEE Transactions on Electron Devices*, *61*(8), 2827–2833. https://doi.org/10.1109/TED.2014.2331707

Kannan, S., Karimi, N., Karri, R., & Sinanoglu, O. (2015). Modeling, Detection, and Diagnosis of Faults in Multilevel Memristor Memories. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(5), 822–834. https://doi.org/10.1109/TCAD.2015.2394434

Kern, J., Henwood, S., Mordido, G., Dupraz, E., Aissa-El-Bey, A., Savaria, Y., & Leduc-Primeau, F. (2022). MemSE: Fast MSE Prediction for Noisy Memristor-Based DNN Accelerators. *Proceeding - IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2022*, 62–65. https://doi.org/10.1109/AICAS54282.2022.9869978

Khaddam-Aljameh, R., Stanisavljevic, M., Fornt Mas, J., Karunaratne, G., Brandli, M., Liu, F., Singh, A., Muller, S. M., Egger, U., Petropoulos, A., Antonakopoulos, T., Brew, K., Choi, S., Ok, I., Lie, F. L., Saulnier, N., Chan, V., Ahsan, I., Narayanan, V., … Eleftheriou, E. (2022). HERMES-Core-A 1.59-TOPS/mm2PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs. *IEEE Journal of Solid-State Circuits*, *57*(4), 1027–1038. https://doi.org/10.1109/JSSC.2022.3140414

KI, R., & R, S. (2022). Memristor based object detection using neural network. *High-Confidence Computing*, *2*(4), 100085. https://doi.org/10.1016/j.hcc.2022.100085

Kim, H., Sah, M. P., Yang, C., & Chua, L. O. (2010). Memristor-based multilevel memory. *2010 12th International Workshop on Cellular Nanoscale Networks and Their Applications, CNNA 2010*, 1–6. https://doi.org/10.1109/cnna.2010.5430320

Kim, S., Du, C., Sheridan, P., Ma, W., Choi, S., & Lu, W. D. (2015). Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity. *Nano Letters*, *15*(3), 2203–2211. https://doi.org/10.1021/acs.nanolett.5b00697

Kim, S., Kim, H.-D., & Choi, S.-J. (2019). Impact of Synaptic Device Variations on Classification Accuracy in a Binarized Neural Network. *Scientific Reports*, *9*(1), 15237. https://doi.org/10.1038/s41598-019-51814-5

Krzysteczko, P., Münchenberger, J., Schäfers, M., Reiss, G., & Thomas, A. (2012). The Memristive Magnetic Tunnel Junction as a Nanoscopic Synapse-Neuron System. *Advanced Materials*, *24*(6), 762–766. https://doi.org/10.1002/adma.201103723

Kuzum, D. (2018). Neuro-Inspired Computing with Resistive Switching Devices [Guest Editorial]. *IEEE Nanotechnology Magazine*, *12*(3), 4–4. https://doi.org/10.1109/MNANO.2018.2849799

Kuzum, D., Jeyasingh, R. G. D., & Wong, H.-S. P. (2011). Energy efficient programming of nanoelectronic synaptic devices for large-scale implementation of associative and temporal sequence learning. *2011 International Electron Devices Meeting*, 30.3.1-30.3.4. https://doi.org/10.1109/IEDM.2011.6131643

Kuzum, D., Jeyasingh, R. G. D., Yu, S., & Wong, H.-S. P. (2012). Low-Energy Robust Neuromorphic Computation Using Synaptic Devices. *IEEE Transactions on Electron*

Devices, *59*(12), 3489–3494. https://doi.org/10.1109/TED.2012.2217146

Kuzum, D., Yu, S., & Philip Wong, H.-S. (2013). Synaptic electronics: materials, devices and applications. *Nanotechnology*, *24*(38), 382001. https://doi.org/10.1088/0957-4484/24/38/382001

Kvatinsky, S., Friedman, E. G., Kolodny, A., & Weiser, U. C. (2013). The Desired Memristor for Circuit Designers. *IEEE Circuits and Systems Magazine*, *13*(2), 17–22. https://doi.org/10.1109/MCAS.2013.2256257

Kvatinsky, S., Satat, G., Wald, N., Friedman, E. G., Kolodny, A., & Weiser, U. C. (2014). Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *22*(10), 2054–2066. https://doi.org/10.1109/TVLSI.2013.2282132

Kwon, O., Kim, S., Agudov, N., Krichigin, A., Mikhaylov, A., Grimaudo, R., Valenti, D., & Spagnolo, B. (2022). Non-volatile memory characteristics of a Ti/HfO2/Pt synaptic device with a crossbar array structure. *Chaos, Solitons & Fractals*, *162*, 112480. https://doi.org/10.1016/j.chaos.2022.112480

Langston, M. H., Greengard, L., & Zorin, D. (2011). A free-space adaptive fmm-based pde solver in three dimensions. *Communications in Applied Mathematics and Computational Science*, *6*(1), 79–122. https://doi.org/10.2140/camcos.2011.6.79

Lashkare, S., Chouhan, S., Chavan, T., Bhat, A., Kumbhare, P., & Ganguly, U. (2018). PCMO RRAM for Integrate-and-Fire Neuron in Spiking Neural Networks. *IEEE Electron Device Letters*, *39*(4), 484–487. https://doi.org/10.1109/LED.2018.2805822

Le Gallo, M., Sebastian, A., Mathis, R., Manica, M., Giefers, H., Tuma, T., Bekas, C., Curioni, A., & Eleftheriou, E. (2018). Mixed-precision in-memory computing. *Nature Electronics*, *1*(4), 246–253. https://doi.org/10.1038/s41928-018-0054-8

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, G., Baek, J., Ren, F., Pearton, S. J., Lee, G., & Kim, J. (2021). Artificial Neuron and Synapse Devices Based on 2D Materials. *Small*, *17*(20), 2100640. https://doi.org/10.1002/smll.202100640

Lee, Y. J., Lee, J., Kim, Y. B., Ayers, J., Volkovskifi, A., Selverston, A., Abarbanel, H., & Rabinovich, M. (2004). Low power real time electronic neuron VLSI design using subthreshold technique. *Proceedings - IEEE International Symposium on Circuits and Systems*, *4*. https://doi.org/10.1109/iscas.2004.1329111

Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., Geman, D., Baggerly, K., & Irizarry, R. A. (2010). Tackling the widespread and critical impact of batch effects in high-throughput data. In *Nature Reviews Genetics* (Vol. 11, Issue 10, pp. 733–739). https://doi.org/10.1038/nrg2825

Li, B., Shan, Y., Hu, M., Wang, Y., Chen, Y., & Yang, H. (2013). Memristor-based approximated computation. *Proceedings of the International Symposium on Low Power Electronics and Design*, *September*, 242–247. https://doi.org/10.1109/ISLPED.2013.6629302

Li, C., Wang, Z., Rao, M., Belkin, D., Song, W., Jiang, H., Yan, P., Li, Y., Lin, P., Hu, M., Ge, N., Strachan, J. P., Barnell, M., Wu, Q., Williams, R. S., Yang, J. J., & Xia, Q.

(2019). Long short-term memory networks in memristor crossbar arrays. *Nature Machine Intelligence*, *1*(1), 49–57. https://doi.org/10.1038/s42256-018-0001-4

Li Deng. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142. https://doi.org/10.1109/MSP.2012.2211477

Li, H., Gao, B., Chen, Z., Zhao, Y., Huang, P., Ye, H., Liu, L., Liu, X., & Kang, J. (2015). A learnable parallel processing architecture towards unity of memory and computing. *Scientific Reports*, *5*(1), 13330. https://doi.org/10.1038/srep13330

Li, S., Li, W., Cook, C., Zhu, C., & Gao, Y. (2019). A fully trainable network with RNN-based pooling. *Neurocomputing*, *338*, 72–82. https://doi.org/10.1016/j.neucom.2019.02.004

Li, S., Zhou, M., Luo, X., & You, Z.-H. (2017). Distributed Winner-Take-All in Dynamic Networks. *IEEE Transactions on Automatic Control*, *62*(2), 577–589. https://doi.org/10.1109/TAC.2016.2578645

Liang, X., Chen, Z., Deng, Y., Liu, D., Liu, X., Huang, Q., & Arai, T. (2023). Field-Controlled Microrobots Fabricated by Photopolymerization. *Cyborg and Bionic Systems*, *4*. https://doi.org/10.34133/cbsystems.0009

Liang, Y., Lu, L., Jin, Y., Xie, J., Huang, R., Zhang, J., & Lin, W. (2022). An Efficient Hardware Design for Accelerating Sparse CNNs With NAS-Based Models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *41*(3), 597–613. https://doi.org/10.1109/TCAD.2021.3066563

Liao, K., Lei, P., Tu, M., Luo, S., Jiang, T., Jie, W., & Hao, J. (2021). Memristor Based on Inorganic and Organic Two-Dimensional Materials: Mechanisms, Performance, and Synaptic Applications. *ACS Applied Materials & Interfaces*, *13*(28), 32606–32623. https://doi.org/10.1021/acsami.1c07665

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*(6), 335–346. https://doi.org/10.1038/s41583-020-0277-3

Lim, S., Bae, J.-H., Eum, J.-H., Lee, S., Kim, C.-H., Kwon, D., Park, B.-G., & Lee, J.-H. (2019). Adaptive learning rule for hardware-based deep neural networks using electronic synapse devices. *Neural Computing and Applications*, *31*(11), 8101–8116. https://doi.org/10.1007/s00521-018-3659-y

Lin, P., Li, C., Wang, Z., Li, Y., Jiang, H., Song, W., Rao, M., Zhuo, Y., Upadhyay, N. K., Barnell, M., Wu, Q., Yang, J. J., & Xia, Q. (2020). Three-dimensional memristor circuits as complex neural networks. *Nature Electronics*, *3*(4), 225–232. https://doi.org/10.1038/s41928-020-0397-9

Lin, R., Shi, G., Qiao, F., Wang, C., & Wu, S. (2023). Research progress and applications of memristor emulator circuits. *Microelectronics Journal*, *133*(January), 105702. https://doi.org/10.1016/j.mejo.2023.105702

Linares-Barranco, B., & Serrano-Gotarredona, T. (2009). Memristance can explain Spike-Time-Dependent-Plasticity in Neural Synapses. *Nature Precedings*. https://doi.org/10.1038/npre.2009.3010.1

Liu, M., Xia, L., Wang, Y., & Chakrabarty, K. (2019). Fault tolerance in neuromorphic

computing systems. *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 216–223. https://doi.org/10.1145/3287624.3288743

Liu, X., & Zeng, Z. (2022). Memristor crossbar architectures for implementing deep neural networks. *Complex and Intelligent Systems*, *8*(2), 787–802. https://doi.org/10.1007/s40747-021-00282-4

Liu, Y., Iu, H. H.-C., & Qian, Y. (2021). Implementation of Hodgkin-Huxley Neuron Model With the Novel Memristive Oscillator. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *68*(8), 2982–2986. https://doi.org/10.1109/TCSII.2021.3066471

Long, X., Lu, C., Su, Y., & Dai, Y. (2023). Machine learning framework for predicting the low cycle fatigue life of lead-free solders. *Engineering Failure Analysis*, *148*, 107228. https://doi.org/10.1016/j.engfailanal.2023.107228

Lu, Y. F., Li, Y., Li, H., Wan, T. Q., Huang, X., He, Y. H., & Miao, X. (2020). Low-Power Artificial Neurons Based on Ag/TiN/HfAlOx/Pt Threshold Switching Memristor for Neuromorphic Computing. *IEEE Electron Device Letters*, *41*(8), 1245–1248. https://doi.org/10.1109/LED.2020.3006581

Lukoševičius, M., & Jaeger, H. (2009a). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149. https://doi.org/10.1016/j.cosrev.2009.03.005

Lukoševičius, M., & Jaeger, H. (2009b). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149. https://doi.org/10.1016/j.cosrev.2009.03.005

Luo, Y., Peng, X., & Yu, S. (2019). MLP+NeuroSimV3.0. *Proceedings of the International Conference on Neuromorphic Systems*, 1–7. https://doi.org/10.1145/3354265.3354266

Lv, Z., Zhou, Y., Han, S. T., & Roy, V. A. L. (2018). From biomaterial-based data storage to bio-inspired artificial synapse. *Materials Today*, *21*(5), 537–552. https://doi.org/10.1016/j.mattod.2017.12.001

Ma, S., Chen, Y., Yang, S., Liu, S., Tang, L., Li, B., & Li, Y. (2023). The Autonomous Pipeline Navigation of a Cockroach Bio-Robot with Enhanced Walking Stimuli. *Cyborg and Bionic Systems*, *4*. https://doi.org/10.34133/cbsystems.0067

Maan, A. K., Jayadevi, D. A., & James, A. P. (2017). A Survey of Memristive Threshold Logic Circuits. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(8), 1734–1746. https://doi.org/10.1109/TNNLS.2016.2547842

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, *4*(3), 161–175. https://doi.org/10.1016/j.dcan.2017.10.002

Mai, H. T., Lieu, Q. X., Kang, J., & Lee, J. (2023). A novel deep unsupervised learning-based framework for optimization of truss structures. *Engineering with Computers*, *39*(4), 2585–2608. https://doi.org/10.1007/s00366-022-01636-3

Manfrinato, V. R., Zhang, L., Su, D., Duan, H., Hobbs, R. G., Stach, E. A., & Berggren, K. K. (2013). Resolution Limits of Electron-Beam Lithography toward the Atomic Scale. *Nano Letters*, *13*(4), 1555–1558. https://doi.org/10.1021/nl304715p

Mao, H., & Dally, W. J. (2016). *N ETWORKS WITH P RUNING , T RAINED Q UANTIZATION*. 1–14.

Mazumder, P., Kang, S. M., & Waser, R. (2012). Memristors: Devices, Models, and Applications [Scanning the Issue]. *Proceedings of the IEEE*, *100*(6), 1911–1919. https://doi.org/10.1109/JPROC.2012.2190812

McCrary, M. B. (1992). Urban multicultural trauma patients. *Asha*, *34*(4).

Mead, C. (1990a). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629–1636. https://doi.org/10.1109/5.58356

Mead, C. (1990b). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629–1636. https://doi.org/10.1109/5.58356

Mead, C. (1990c). Neuromorphic Electronic Systems. *Proceedings of the IEEE*, *78*(10), 1629–1636. https://doi.org/10.1109/5.58356

Mehonic, A., & Kenyon, A. J. (2016). Emulating the Electrical Activity of the Neuron Using a Silicon Oxide RRAM Cell. *Frontiers in Neuroscience*, *10*. https://doi.org/10.3389/fnins.2016.00057

Mehonic, A., Sebastian, A., Rajendran, B., Simeone, O., Vasilaki, E., & Kenyon, A. J. (2020). Memristors—From In-Memory Computing, Deep Learning Acceleration, and Spiking Neural Networks to the Future of Neuromorphic and Bio-Inspired Computing. *Advanced Intelligent Systems*, *2*(11). https://doi.org/10.1002/aisy.202000085

Merced-Grafals, E. J., Dávila, N., Ge, N., Williams, R. S., & Strachan, J. P. (2016). Repeatable, accurate, and high speed multi-level programming of memristor 1T1R arrays for power efficient analog computing applications. *Nanotechnology*, *27*(36), 365202. https://doi.org/10.1088/0957-4484/27/36/365202

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. S. (2011). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 1–4. https://doi.org/10.1109/CICC.2011.6055294

Meuffels, P., & Soni, R. (2012). *Fundamental Issues and Problems in the Realization of Memristors*. *2012*(version 01), 1–14. http://arxiv.org/abs/1207.7319

Miyake, R., Nagata, Z., Adachi, K., Hayashi, Y., Tohei, T., & Sakai, A. (2022). Versatile Functionality of Four-Terminal TiO 2– x Memristive Devices as Artificial Synapses for Neuromorphic Computing. *ACS Applied Electronic Materials*, *4*(5), 2326–2336. https://doi.org/10.1021/acsaelm.2c00161

Mizrahi, A., Hirtzlin, T., Fukushima, A., Kubota, H., Yuasa, S., Grollier, J., & Querlioz, D. (2018). Neural-like computing with populations of superparamagnetic basis functions. *Nature Communications*, *9*(1). https://doi.org/10.1038/s41467-018-03963-w

Moore, G. M. (1965). Cramming more components onto integrated circuits With unit cost. *Electronics*, *38*(8), 114. https://newsroom.intel.com/wp-content/uploads/sites/11/2018/05/moores-law-electronics.pdf

Musisi-Nkambwe, M., Afshari, S., Barnaby, H., Kozicki, M., & Sanchez Esqueda, I. (2021). The viability of analog-based accelerators for neuromorphic computing: a survey. *Neuromorphic Computing and Engineering*, *1*(1), 012001. https://doi.org/10.1088/2634-

4386/ac0242

Mutlu, O., Ghose, S., Gómez-Luna, J., & Ausavarungnirun, R. (2019a). Processing data where it makes sense: Enabling in-memory computation. *Microprocessors and Microsystems*, *67*, 28–41. https://doi.org/10.1016/j.micpro.2019.01.009

Mutlu, O., Ghose, S., Gómez-Luna, J., & Ausavarungnirun, R. (2019b). Processing data where it makes sense: Enabling in-memory computation. *Microprocessors and Microsystems*, *67*, 28–41. https://doi.org/10.1016/j.micpro.2019.01.009

Narayanan, P., Ambrogio, S., Okazaki, A., Hosokawa, K., Tsai, H., Nomura, A., Yasuda, T., Mackin, C., Lewis, S. C., Friz, A., Ishii, M., Kohda, Y., Mori, H., Spoon, K., Khaddam-Aljameh, R., Saulnier, N., Bergendahl, M., Demarest, J., Brew, K. W., … Burr, G. W. (2021). Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format. *IEEE Transactions on Electron Devices*, *68*(12), 6629–6636. https://doi.org/10.1109/TED.2021.3115993

Nawrocki, R. A., Voyles, R. M., & Shaheen, S. E. (2016). A Mini Review of Neuromorphic Architectures and Implementations. *IEEE Transactions on Electron Devices*, *63*(10), 3819–3829. https://doi.org/10.1109/TED.2016.2598413

Niu, D., Chen, Y., & Xie, Y. (2010). Low-power dual-element memristor based memory design. *Proceedings of the International Symposium on Low Power Electronics and Design*, 25–30. https://doi.org/10.1145/1840845.1840851

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. http://arxiv.org/abs/1811.03378

Oh, S., Shi, Y., del Valle, J., Salev, P., Lu, Y., Huang, Z., Kalcheim, Y., Schuller, I. K., & Kuzum, D. (2021). Energy-efficient Mott activation neuron for full-hardware implementation of neural networks. *Nature Nanotechnology*, *16*(6), 680–687. https://doi.org/10.1038/s41565-021-00874-8

Oli-Uz-Zaman, M., Khan, S. A., Yuan, G., Liao, Z., Fu, J., Ding, C., Wang, Y., & Wang, J. (2022). Mapping Transformation Enabled High-Performance and Low-Energy Memristor-Based DNNs. *Journal of Low Power Electronics and Applications*, *12*(1), 10. https://doi.org/10.3390/jlpea12010010

Onen, M., Butters, B. A., Toomey, E., Gokmen, T., & Berggren, K. K. (2020). Design and characterization of superconducting nanowire-based processors for acceleration of deep neural network training. *Nanotechnology*, *31*(2), 025204. https://doi.org/10.1088/1361-6528/ab47bc

Palit, I., Sedighi, B., Horvath, A., Hu, X. S., Nahas, J., & Niemier, M. (2014). Impact of steep-slope transistors on non-von Neumann architectures: CNN case study. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 1–6. https://doi.org/10.7873/DATE.2014.150

Pantazi, A., Woźniak, S., Tuma, T., & Eleftheriou, E. (2016). All-memristive neuromorphic computing with level-tuned neurons. *Nanotechnology*, *27*(35), 355205. https://doi.org/10.1088/0957-4484/27/35/355205

Pedretti, G., Milo, V., Ambrogio, S., Carboni, R., Bianchi, S., Calderoni, A., Ramaswamy, N.,

Spinelli, A. S., & Ielmini, D. (2017). Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Scientific Reports*, *7*(1), 5288. https://doi.org/10.1038/s41598-017-05480-0

Pershin, Y. V., & Di Ventra, M. (2010). Experimental demonstration of associative memory with memristive neural networks. *Neural Networks*, *23*(7), 881–886. https://doi.org/10.1016/j.neunet.2010.05.001

Pershin, Y. V., & Di Ventra, M. (2012). Neuromorphic, Digital, and Quantum Computation With Memory Circuit Elements. *Proceedings of the IEEE*, *100*(6), 2071–2080. https://doi.org/10.1109/JPROC.2011.2166369

Pickett, M. D., Medeiros-Ribeiro, G., & Williams, R. S. (2013). A scalable neuristor built with Mott memristors. *Nature Materials*, *12*(2), 114–117. https://doi.org/10.1038/nmat3510

Prezioso, M., Merrikh-Bayat, F., Hoskins, B. D., Adam, G. C., Likharev, K. K., & Strukov, D. B. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, *521*(7550), 61–64. https://doi.org/10.1038/nature14441

Prezioso, M., Merrikh Bayat, F., Hoskins, B., Likharev, K., & Strukov, D. (2016). Self-Adaptive Spike-Time-Dependent Plasticity of Metal-Oxide Memristors. *Scientific Reports*, *6*(1), 21331. https://doi.org/10.1038/srep21331

Prihatno, A. T., Nurcahyanto, H., Ahmed, M. F., Rahman, M. H., Alam, M. M., & Jang, Y. M. (2021). Forecasting PM2.5 Concentration Using a Single-Dense Layer BiLSTM Method. *Electronics*, *10*(15), 1808. https://doi.org/10.3390/electronics10151808

Qin, Y.-F., Bao, H., Wang, F., Chen, J., Li, Y., & Miao, X.-S. (2020). Recent Progress on Memristive Convolutional Neural Networks for Edge Intelligence. *Advanced Intelligent Systems*, *2*(11). https://doi.org/10.1002/aisy.202000114

Radack, D. J., & Zolper, J. C. (2008). A Future of Integrated Electronics: Moving Off the Roadmap. *Proceedings of the IEEE*, *96*(2), 198–200. https://doi.org/10.1109/JPROC.2007.911049

Rahimi Azghadi, M., Chen, Y.-C., Eshraghian, J. K., Chen, J., Lin, C.-Y., Amirsoleimani, A., Mehonic, A., Kenyon, A. J., Fowler, B., Lee, J. C., & Chang, Y.-F. (2020). Complementary Metal-Oxide Semiconductor and Memristive Hardware for Neuromorphic Computing. *Advanced Intelligent Systems*, *2*(5). https://doi.org/10.1002/aisy.201900189

Rajendran, B., Sebastian, A., Schmuker, M., Srinivasa, N., & Eleftheriou, E. (2019). *Low-Power Neuromorphic Hardware for Signal Processing Applications*. http://arxiv.org/abs/1901.03690

Ranjan, R., Ponce, P. M., Kankuppe, A., John, B., Saleh, L. A., Schroeder, D., & Krautschneider, W. H. (2016). Programmable memristor emulator ASIC for biologically inspired memristive learning. *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, *5*, 261–264. https://doi.org/10.1109/TSP.2016.7760874

Ren, Y., Yi, K., Zhang, Y., & Tong, N. (2022). A Memristor-Based DNN Crossbar Array for Iterative Network Pruning and Quantization. In *Citation:. Journal Not Specified* (Vol. 2022). https://doi.org/

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., & Suter, B. W. (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, *1*(4), 296–298. https://doi.org/10.1109/72.80266

Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. http://arxiv.org/abs/1609.04747

S. Raj, J., & Ananthi J, V. (2019). Recurrent Neural Networks and Nonlinear Prediction in Support Vector Machines. *Journal of Soft Computing Paradigm*, *2019*(1), 33–40. https://doi.org/10.36548/jscp.2019.1.004

Sah, M. P., Yang, C., Kim, H., Muthuswamy, B., Jevtic, J., & Chua, L. (2015). A generic model of memristors with parasitic components. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *62*(3), 891–898. https://doi.org/10.1109/TCSI.2014.2373674

Saïghi, S., Mayr, C. G., Serrano-Gotarredona, T., Schmidt, H., Lecerf, G., Tomas, J., Grollier, J., Boyn, S., Vincent, A. F., Querlioz, D., La Barbera, S., Alibart, F., Vuillaume, D., Bichler, O., Gamrat, C., & Linares-Barranco, B. (2015). Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience*, *9*. https://doi.org/10.3389/fnins.2015.00051

Sarker, I. H., Hoque, M. M., Uddin, M. K., & Alsanoosy, T. (2021). Mobile Data Science and Intelligent Apps: Concepts, AI-Based Modeling and Research Directions. *Mobile Networks and Applications*, *26*(1), 285–303. https://doi.org/10.1007/s11036-020-01650-z

Saxena, V., Wu, X., & Zhu, K. (2018). Energy-Efficient CMOS Memristive Synapses for Mixed-Signal Neuromorphic System-on-a-Chip. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. https://doi.org/10.1109/ISCAS.2018.8351766

Schmidhuber, J. (2015). Deep Learning. *Scholarpedia*, *10*(11), 32832. https://doi.org/10.4249/scholarpedia.32832

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., & Plank, J. S. (2017a). *A Survey of Neuromorphic Computing and Neural Networks in Hardware*. http://arxiv.org/abs/1705.06963

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., & Plank, J. S. (2017b). *A Survey of Neuromorphic Computing and Neural Networks in Hardware*. 1–88.

Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R., & Eleftheriou, E. (2020a). Memory devices and applications for in-memory computing. *Nature Nanotechnology*, *15*(7), 529–544. https://doi.org/10.1038/s41565-020-0655-z

Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R., & Eleftheriou, E. (2020b). Memory devices and applications for in-memory computing. *Nature Nanotechnology*, *15*(7), 529–544. https://doi.org/10.1038/s41565-020-0655-z

Sejnowski, T. J., Koch, C., & Churchland, P. S. (1988). Computational Neuroscience. *Science*, *241*(4871), 1299–1306. https://doi.org/10.1126/science.3045969

Seo, J., Brezzo, B., Liu, Y., Parker, B. D., Esser, S. K., Montoye, R. K., Rajendran, B.,

Tierno, J. A., Chang, L., Modha, D. S., & Friedman, D. J. (2011). A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 1–4. https://doi.org/10.1109/CICC.2011.6055293

Seo, S., Kang, B.-S., Lee, J.-J., Ryu, H.-J., Kim, S., Kim, H., Oh, S., Shim, J., Heo, K., Oh, S., & Park, J.-H. (2020). Artificial van der Waals hybrid synapse and its application to acoustic pattern recognition. *Nature Communications*, *11*(1), 3936. https://doi.org/10.1038/s41467-020-17849-3

Serb, A., Bill, J., Khiat, A., Berdan, R., Legenstein, R., & Prodromakis, T. (2016). Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses. *Nature Communications*, *7*. https://doi.org/10.1038/ncomms12611

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., & Srikumar, V. (2016). ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 14–26. https://doi.org/10.1109/ISCA.2016.12

Shamsi, J., Amirsoleimani, A., Mirzakuchaki, S., & Ahmadi, M. (2017). Modular neuron comprises of memristor-based synapse. *Neural Computing and Applications*, *28*(1), 1–11. https://doi.org/10.1007/s00521-015-2047-0

Shamsi, J., Mohammadi, K., & Shokouhi, S. B. (2018). A Hardware Architecture for Columnar-Organized Memory Based on CMOS Neuron and Memristor Crossbar Arrays. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *26*(12), 2795–2805. https://doi.org/10.1109/TVLSI.2018.2815025

Sharma, S., Sharma, S., & Anidhya, A. (2020). Understanding Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, *4*(12), 310–316.

Shaw, R. F. (1950). Arithmetic Operations in a Binary Computer. *Review of Scientific Instruments*, *21*(8), 687–693. https://doi.org/10.1063/1.1745692

Sheliakina, M., Mostert, A. B., & Meredith, P. (2018). An all-solid-state biocompatible ion-to-electron transducer for bioelectronics. *Materials Horizons*, *5*(2), 256–263. https://doi.org/10.1039/C7MH00831G

Shi, J., Ha, S. D., Zhou, Y., Schoofs, F., & Ramanathan, S. (2013). A correlated nickelate synaptic transistor. *Nature Communications*, *4*, 1–9. https://doi.org/10.1038/ncomms3676

Shi, Y., Nguyen, L., Oh, S., Liu, X., & Kuzum, D. (2019). A Soft-Pruning Method Applied During Training of Spiking Neural Networks for In-memory Computing Applications. *Frontiers in Neuroscience*, *13*(APR), 1–13. https://doi.org/10.3389/fnins.2019.00405

Shim, W., Seo, J., & Yu, S. (2020). Two-step write–verify scheme and impact of the read noise in multilevel RRAM-based inference engine. *Semiconductor Science and Technology*, *35*(11), 115026. https://doi.org/10.1088/1361-6641/abb842

Shvetsov, B. S., Minnekhanov, A. A., Emelyanov, A. V, Ilyasov, A. I., Grishchenko, Y. V, Zanaveskin, M. L., Nesmelov, A. A., Streltsov, D. R., Patsaev, T. D., Vasiliev, A. L., Rylkov, V. V, & Demin, V. A. (2022). lene-based memristive crossbar structures with

multilevel resistive switching for neuromorphic computingPary. *Nanotechnology*, *33*(25), 255201. https://doi.org/10.1088/1361-6528/ac5cfe

Shymkovych, V., Telenyk, S., & Kravets, P. (2021). Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. *Neural Computing and Applications*, *33*(15), 9467–9479. https://doi.org/10.1007/s00521-021-05706-3

Soneson, C., Gerster, S., & Delorenzi, M. (2014). Batch effect confounding leads to strong bias in performance estimates obtained by cross-validation. *PLoS ONE*, *9*(6). https://doi.org/10.1371/journal.pone.0100335

Sourikopoulos, I., Hedayat, S., Loyez, C., Danneville, F., Hoel, V., Mercier, E., & Cappy, A. (2017). A 4-fJ/spike artificial neuron in 65 nm CMOS technology. *Frontiers in Neuroscience*, *11*(MAR). https://doi.org/10.3389/fnins.2017.00123

Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). The missing memristor found. *Nature*, *453*(7191), 80–83. https://doi.org/10.1038/nature06932

Sun, S., Cao, Z., Zhu, H., & Zhao, J. (2020). A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Transactions on Cybernetics*, *50*(8), 3668–3681. https://doi.org/10.1109/TCYB.2019.2950779

Sung, C., Hwang, H., & Yoo, I. K. (2018a). Perspective: A review on memristive hardware for neuromorphic computation. *Journal of Applied Physics*, *124*(15). https://doi.org/10.1063/1.5037835

Sung, C., Hwang, H., & Yoo, I. K. (2018b). Perspective: A review on memristive hardware for neuromorphic computation. *Journal of Applied Physics*, *124*(15), 151903. https://doi.org/10.1063/1.5037835

Sung, C., Hwang, H., & Yoo, I. K. (2018c). Perspective: A review on memristive hardware for neuromorphic computation. *Journal of Applied Physics*, *124*(15), 151903. https://doi.org/10.1063/1.5037835

Sung, S. H., Kim, T. J., Shin, H., Namkung, H., Im, T. H., Wang, H. S., & Lee, K. J. (2021). Memory-centric neuromorphic computing for unstructured data processing. *Nano Research*, *14*(9), 3126–3142. https://doi.org/10.1007/s12274-021-3452-6

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, *105*(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

Taur, Y. (2002). CMOS design near the limit of scaling. *IBM Journal of Research and Development*, *46*(2.3), 213–222. https://doi.org/10.1147/rd.462.0213

Thimm, G., & Fiesler, E. (1997). High-order and multilayer perceptron initialization. *IEEE Transactions on Neural Networks*, *8*(2), 349–359. https://doi.org/10.1109/72.557673

Thomas, A. (2013). Memristor-based neural networks. *Journal of Physics D: Applied Physics*, *46*(9), 093001. https://doi.org/10.1088/0022-3727/46/9/093001

Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M., & Burr, G. W. (2018a). Recent progress in analog memory-based accelerators for deep learning. *Journal of Physics D: Applied Physics*, *51*(28), 283001. https://doi.org/10.1088/1361-6463/aac8a5

Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M., & Burr, G. W. (2018b). Recent progress in analog memory-based accelerators for deep learning. In *Journal of Physics*

D: Applied Physics (Vol. 51, Issue 28). Institute of Physics Publishing. https://doi.org/10.1088/1361-6463/aac8a5

Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A., & Eleftheriou, E. (2016). Stochastic phase-change neurons. *Nature Nanotechnology*, *11*(8), 693–699. https://doi.org/10.1038/nnano.2016.70

Turian, J., Bergstra, J., & Bengio, Y. (2009). Quadratic features and deep architectures for chunking. *NAACL-HLT 2009 - Human Language Technologies: 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Short Papers*, *June*, 245–248. https://doi.org/10.3115/1620853.1620921

Van De Burgt, Y., Lubberman, E., Fuller, E. J., Keene, S. T., Faria, G. C., Agarwal, S., Marinella, M. J., Alec Talin, A., & Salleo, A. (2017). A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature Materials*, *16*(4), 414–418. https://doi.org/10.1038/NMAT4856

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Vaughan, O. (2023). A history of memristors in five covers. *Nature Electronics*, *6*(1), 7–7. https://doi.org/10.1038/s41928-023-00923-1

von Neumann, J. (1993). First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, *15*(4), 27–75. https://doi.org/10.1109/85.238389

Wan, Q., Sharbati, M. T., Erickson, J. R., Du, Y., & Xiong, F. (2019a). Emerging Artificial Synaptic Devices for Neuromorphic Computing. *Advanced Materials Technologies*, *4*(4), 1900037. https://doi.org/10.1002/admt.201900037

Wan, Q., Sharbati, M. T., Erickson, J. R., Du, Y., & Xiong, F. (2019b). Emerging Artificial Synaptic Devices for Neuromorphic Computing. *Advanced Materials Technologies*, *4*(4). https://doi.org/10.1002/admt.201900037

Wan, W., Kubendran, R., Schaefer, C., Eryilmaz, S. B., Zhang, W., Wu, D., Deiss, S., Raina, P., Qian, H., Gao, B., Joshi, S., Wu, H., Wong, H. S. P., & Cauwenberghs, G. (2022). A compute-in-memory chip based on resistive random-access memory. *Nature*, *608*(7923), 504–512. https://doi.org/10.1038/s41586-022-04992-8

Wang, F. Y. (2008). *Memristor for Introductory Physics*. *1*, 1–4. http://arxiv.org/abs/0808.0286

Wang, H., Zhao, Q., Ni, Z., Li, Q., Liu, H., Yang, Y., Wang, L., Ran, Y., Guo, Y., Hu, W., & Liu, Y. (2018a). A Ferroelectric/Electrochemical Modulated Organic Synapse for Ultraflexible, Artificial Visual-Perception System. *Advanced Materials*, *30*(46), 1803961. https://doi.org/10.1002/adma.201803961

Wang, H., Zhao, Q., Ni, Z., Li, Q., Liu, H., Yang, Y., Wang, L., Ran, Y., Guo, Y., Hu, W., & Liu, Y. (2018b). A Ferroelectric/Electrochemical Modulated Organic Synapse for Ultraflexible, Artificial Visual-Perception System. *Advanced Materials*, *30*(46), 1803961. https://doi.org/10.1002/adma.201803961

Wang, Q., Zhang, S., Ji, X., & Ran, F. (2020). High rejection performance ultrafiltration membrane with ultrathin dense layer fabricated by the movement and dissolution of metal–organic frameworks. *New Journal of Chemistry*, *44*(32), 13745–13754.

https://doi.org/10.1039/D0NJ02700F

Wang, Y. E., Wei, G.-Y., & Brooks, D. (2019). *Benchmarking TPU, GPU, and CPU Platforms for Deep Learning*. http://arxiv.org/abs/1907.10701

Wang, Z., Joshi, S., Savel'ev, S. E., Jiang, H., Midya, R., Lin, P., Hu, M., Ge, N., Strachan, J. P., Li, Z., Wu, Q., Barnell, M., Li, G.-L., Xin, H. L., Williams, R. S., Xia, Q., & Yang, J. J. (2017). Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nature Materials*, *16*(1), 101–108. https://doi.org/10.1038/nmat4756

Wang, Z., Joshi, S., Savel'Ev, S., Song, W., Midya, R., Li, Y., Rao, M., Yan, P., Asapu, S., Zhuo, Y., Jiang, H., Lin, P., Li, C., Yoon, J. H., Upadhyay, N. K., Zhang, J., Hu, M., Strachan, J. P., Barnell, M., … Yang, J. J. (2018). Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, *1*(2), 137–145. https://doi.org/10.1038/s41928-018-0023-2

Wang, Z., Li, C., Song, W., Rao, M., Belkin, D., Li, Y., Yan, P., Jiang, H., Lin, P., Hu, M., Strachan, J. P., Ge, N., Barnell, M., Wu, Q., Barto, A. G., Qiu, Q., Williams, R. S., Xia, Q., & Yang, J. J. (2019). Reinforcement learning with analogue memristor arrays. *Nature Electronics*, *2*(3), 115–124. https://doi.org/10.1038/s41928-019-0221-6

Wang, Z., Wu, H., Burr, G. W., Hwang, C. S., Wang, K. L., Xia, Q., & Yang, J. J. (2020). Resistive switching materials for information processing. *Nature Reviews Materials*, *5*(3), 173–195. https://doi.org/10.1038/s41578-019-0159-3

Wijekoon, J. H. B., & Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behaviour. *Neural Networks*, *21*(2–3), 524–534. https://doi.org/10.1016/j.neunet.2007.12.037

Wijesinghe, P., Ankit, A., Sengupta, A., & Roy, K. (2018). An All-Memristor Deep Spiking Neural Computing System: A Step Toward Realizing the Low-Power Stochastic Brain. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(5), 345–358. https://doi.org/10.1109/TETCI.2018.2829924

Wong, H. S. P., Lee, H. Y., Yu, S., Chen, Y. S., Wu, Y., Chen, P. S., Lee, B., Chen, F. T., & Tsai, M. J. (2012). Metal-oxide RRAM. *Proceedings of the IEEE*, *100*(6), 1951–1970. https://doi.org/10.1109/JPROC.2012.2190369

Wong, H. S. P., Raoux, S., Kim, S., Liang, J., Reifenberg, J. P., Rajendran, B., Asheghi, M., & Goodson, K. E. (2010). Phase change memory. *Proceedings of the IEEE*, *98*(12), 2201–2227. https://doi.org/10.1109/JPROC.2010.2070050

Woo, J., Moon, K., Song, J., Lee, S., Kwak, M., Park, J., & Hwang, H. (2016). Improved Synaptic Behavior Under Identical Pulses Using AlO x /HfO 2 Bilayer RRAM Array for Neuromorphic Systems. *IEEE Electron Device Letters*, *37*(8), 994–997. https://doi.org/10.1109/LED.2016.2582859

Xia, Z., Chen, J., Huang, Q., Luo, J., & Hu, J. (2021). Neural Synaptic Plasticity-Inspired Computing: A High Computing Efficient Deep Convolutional Neural Network Accelerator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *68*(2), 728–740. https://doi.org/10.1109/TCSI.2020.3039346

Xiao, T. P., Bennett, C. H., Feinberg, B., Agarwal, S., & Marinella, M. J. (2020). Analog architectures for neural network acceleration based on non-volatile memory. *Applied Physics Reviews*, *7*(3). https://doi.org/10.1063/1.5143815

Xinyu Wu, Saxena, V., & Kehan Zhu. (2015). A CMOS spiking neuron for dense memristor-synapse connectivity for brain-inspired computing. *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–6. https://doi.org/10.1109/IJCNN.2015.7280819

Xiu, L. (2019). Time Moore: Exploiting Moore's Law From The Perspective of Time. *IEEE Solid-State Circuits Magazine*, *11*(1), 39–55. https://doi.org/10.1109/MSSC.2018.2882285

Xu, J., Li, Z., Du, B., Zhang, M., & Liu, J. (2020). Reluplex made more practical: Leaky ReLU. *Proceedings - IEEE Symposium on Computers and Communications*, *2020-July*. https://doi.org/10.1109/ISCC50000.2020.9219587

Xu, W., Cho, H., Kim, Y.-H., Kim, Y.-T., Wolf, C., Park, C.-G., & Lee, T.-W. (2016). Organometal Halide Perovskite Artificial Synapses. *Advanced Materials*, *28*(28), 5916–5922. https://doi.org/10.1002/adma.201506363

Yakopcic, C., Taha, T. M., Subramanyam, G., & Rogers, S. (2011). Multiple memristor read and write circuit for neuromorphic applications. *The 2011 International Joint Conference on Neural Networks*, 2676–2682. https://doi.org/10.1109/IJCNN.2011.6033569

Yan, X., Zhang, L., Chen, H., Li, X., Wang, J., Liu, Q., Lu, C., Chen, J., Wu, H., & Zhou, P. (2018). Graphene Oxide Quantum Dots Based Memristors with Progressive Conduction Tuning for Artificial Synaptic Learning. *Advanced Functional Materials*, *28*(40), 1803728. https://doi.org/10.1002/adfm.201803728

Yang, J., Liu, Y., Qian, M., Guan, C., & Yuan, X. (2019). Information Extraction from Electronic Medical Records Using Multitask Recurrent Neural Network with Contextual Word Embedding. *Applied Sciences*, *9*(18), 3658. https://doi.org/10.3390/app9183658

Yang, T.-H., Li, K.-X., Chiang, Y.-N., Lin, W.-Y., Lin, H.-T., & Chang, M.-F. (2018). A 28nm 32Kb embedded 2T2MTJ STT-MRAM macro with 1.3ns read-access time for fast and reliable read applications. *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 482–484. https://doi.org/10.1109/ISSCC.2018.8310394

Yao, P., Wu, H., Gao, B., Eryilmaz, S. B., Huang, X., Zhang, W., Zhang, Q., Deng, N., Shi, L., Wong, H. S. P., & Qian, H. (2017). Face classification using electronic synapses. *Nature Communications*, *8*(May), 1–8. https://doi.org/10.1038/ncomms15199

Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J. J., & Qian, H. (2020). Fully hardware-implemented memristor convolutional neural network. *Nature*, *577*(7792), 641–646. https://doi.org/10.1038/s41586-020-1942-4

Ye, L., Gao, Z., Fu, J., Ren, W., Yang, C., Wen, J., Wan, X., Ren, Q., Gu, S., Liu, X., Lian, X., & Wang, L. (2022). Overview of Memristor-Based Neural Network Design and Applications. *Frontiers in Physics*, *10*(July), 1–27. https://doi.org/10.3389/fphy.2022.839243

Yoo, I. K., Kang, B. S., Park, Y. D., Lee, M. J., & Park, Y. (2008). Interpretation of nanoscale conducting paths and their control in nickel oxide (NiO) thin films. *Applied Physics Letters*, *92*(20), 202112. https://doi.org/10.1063/1.2936087

Yu, H., Wei, H., Gong, J., Han, H., Ma, M., Wang, Y., & Xu, W. (2021). Evolution of Bio-Inspired Artificial Synapses: Materials, Structures, and Mechanisms. *Small*, *17*(9), 2000041. https://doi.org/10.1002/smll.202000041

Yu, S., Wu, Y., Jeyasingh, R., Kuzum, D., & Wong, H.-S. P. (2011a). An Electronic Synapse

Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation. *IEEE Transactions on Electron Devices*, *58*(8), 2729–2737. https://doi.org/10.1109/TED.2011.2147791

Yu, S., Wu, Y., Jeyasingh, R., Kuzum, D., & Wong, H.-S. P. (2011b). An Electronic Synapse Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation. *IEEE Transactions on Electron Devices*, *58*(8), 2729–2737. https://doi.org/10.1109/TED.2011.2147791

Zahari, F., Hansen, M., Mussenbrock, T., Ziegler, M., & Kohlstedt, H. (2015). Pattern recognition with TiOx-based memristive devices. *AIMS Materials Science*, *2*(3), 203–216. https://doi.org/10.3934/matersci.2015.3.203

Zanotti, T., Puglisi, F. M., & Pavan, P. (2020). Smart Logic-in-Memory Architecture for Low-Power Non-Von Neumann Computing. *IEEE Journal of the Electron Devices Society*, *8*(March), 757–764. https://doi.org/10.1109/JEDS.2020.2987402

Zanotti, T., Puglisi, F. M., & Pavan, P. (2021). Energy-Efficient Non-Von Neumann Computing Architecture Supporting Multiple Computing Paradigms for Logic and Binarized Neural Networks. *Journal of Low Power Electronics and Applications*, *11*(3), 29. https://doi.org/10.3390/jlpea11030029

Zeiler, M. D. (2012). *ADADELTA: An Adaptive Learning Rate Method*. http://arxiv.org/abs/1212.5701

Zeng, J., Chen, X., Liu, S., Chen, Q., & Liu, G. (2023). Organic Memristor with Synaptic Plasticity for Neuromorphic Computing Applications. *Nanomaterials*, *13*(5), 803. https://doi.org/10.3390/nano13050803

Zhang, J. J., Sun, H. J., Li, Y., Wang, Q., Xu, X. H., & Miao, X. S. (2013). AgInSbTe memristor with gradual resistance tuning. *Applied Physics Letters*, *102*(18), 183513. https://doi.org/10.1063/1.4804983

Zhang, J., & Liao, X. (2017). Synchronization and chaos in coupled memristor-based FitzHugh-Nagumo circuits with memristor synapse. *AEU - International Journal of Electronics and Communications*, *75*, 82–90. https://doi.org/10.1016/j.aeue.2017.03.003

Zhang, J., & Zong, C. (2015). Deep Neural Networks in Machine Translation: An Overview. *IEEE Intelligent Systems*, *30*(5), 16–25. https://doi.org/10.1109/MIS.2015.69

Zhang, W., Yao, P., Gao, B., Liu, Q., Wu, D., Zhang, Q., Li, Y., Qin, Q., Li, J., Zhu, Z., Cai, Y., Wu, D., Tang, J., Qian, H., Wang, Y., & Wu, H. (2023). Edge learning using a fully integrated neuro-inspired memristor chip. *Science (New York, N.Y.)*, *381*(6663), 1205–1211. https://doi.org/10.1126/science.ade3483

Zhang, X., Wang, W., Liu, Q., Zhao, X., Wei, J., Cao, R., Yao, Z., Zhu, X., Zhang, F., Lv, H., Long, S., & Liu, M. (2018). An Artificial Neuron Based on a Threshold Switching Memristor. *IEEE Electron Device Letters*, *39*(2), 308–311. https://doi.org/10.1109/LED.2017.2782752

Zheng, N., & Mazumder, P. (2018). Online Supervised Learning for Hardware-Based Multilayer Spiking Neural Networks Through the Modulation of Weight-Dependent Spike-Timing-Dependent Plasticity. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(9), 4287–4302. https://doi.org/10.1109/TNNLS.2017.2761335

Zidan, M. A., Strachan, J. P., & Lu, W. D. (2018). The future of electronics based on

memristive systems. *Nature Electronics*, *1*(1), 22–29. https://doi.org/10.1038/s41928-017-0006-8

Zyarah, A. M., Soures, N., Hays, L., Jacobs-Gedrim, R. B., Agarwal, S., Marinella, M., & Kudithipudi, D. (2017). Ziksa: On-chip learning accelerator with memristor crossbars for multilevel neural networks. *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–4. https://doi.org/10.1109/ISCAS.2017.8050531

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı:	Baki GÖKGÖZ
Doğum tarihi:	
Doğum Yeri:	
Uyruğu:	
Adres:	
Tel:	
E-mail:	
Eğitim	
Lise:	Tekirdağ İmam Hatip Lisesi
Lisans:	Uluslararası Kıbrıs Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü (2009)
Yüksek lisans:	Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı (2016)
Doktora:	Atatürk Üniversitesi
Yabancı Dil Bilgisi	
İngilizce:	İyi

Tezden Üretilmiş Yayınlar

1. **B. Gökgöz**, F. Gül, and T. Aydin, ‘‘An overview memristor based hardware accelerators for deep neural network,’’ Concurrency Comput., Pract. Exper., vol. 36, no. 9, pp. 1–22, Apr. 2024, doi: 10.1002/cpe.7997.
2. **B. GÖKGÖZ**, T. AYDIN, and F. GÜL, “Optimizing Memristor-Based Synaptic Devices for Enhanced Energy Efficiency and Accuracy in Neuromorphic Machine Learning,” *IEEE Access*, vol. 12, pp. 154401–154417, Nov. 2024**.**