FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

İŞBİRLİKÇİ ZEKA YAKLAŞIMIYLA ÇOK GÖREVLİ ÖĞRENMEDE DİNAMİK BÖLME NOKTASI HESAPLAMA

YÜKSEK LİSANS TEZİ

MUHAMMED FARUK ŞAHİN

İSTANBUL, 2025

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

İŞBİRLİKÇİ ZEKA YAKLAŞIMIYLA ÇOK GÖREVLİ ÖĞRENMEDE DİNAMİK BÖLME NOKTASI HESAPLAMA

YÜKSEK LİSANS TEZİ

MUHAMMED FARUK ŞAHİN
(220221003)

Danışman
(Prof. Dr. Ferzat Anka)

İSTANBUL, 2025

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
TEZ ONAY FORMU

24/06/2025

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜNE

BİLGİSAYAR MÜHENDİSLİĞİ Anabilim/Anasanat Dalı BİLGİSAYAR MÜHENDİSLİĞİ programı öğrencisi 220221003 numaralı Muhammed Faruk Şahin.'nın hazırladığı "COLLABORATIVE INTELLIGENCE İLE MULTI-TASK LEARNING UYGULANMASINDA DİNAMİK SPLIT NOKTA HESAPLANMASI"konulu Yüksek Lisans/Doktora/Sanatta Yeterlik tezi ile ilgili Tez Savunma Sınavı, 24/06/2025 Salı. günü saat 11:30'da yapılmış, sorulara alınan cevaplar sonunda adayın tezinin **Kabulüne ~~Oy Çokluğu~~/Oy Birliği** ile karar verilmiştir.

Tez adı değişikliği yapılması halinde: Tez adının İşbirlikçi Zeka Yaklaşımıyla Çok Görevli Öğrenmede Dinamik Bölme Noktası Hesaplama şeklinde değiştirilmesi uygundur.

	Jüri Üyesi	Karar
1.	Prof. Dr.Ferzat ANKA (Danışman)	Kabul
2.	Dr. Öğr. Üyesi Sultan ZEYBEK	Kabul
3.	Dr. Öğr. Üyesi Sajjad Nematzadeh MIANDOAB	Kabul
4.	.....	.....
5.	.....	.....
6.	(İkinci Danışman)*.....	.....

**2. Danışman varsa doldurulması gerekmektedir.*

ETİK BİLDİRİM

Bu tezin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bağlı olduğum üniversite veya bir başka üniversitedeki başka bir çalışma olarak sunulmadığını beyan ederim.

Muhammed Faruk Şahin

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, akademik bilgi ve bilimsel üslubuyla yol gösteren, bilimsel düşünce yapımı şekillendiren danışman hocam Prof. Dr. Ferzat ANKA'ya ve lisans hayatımdan beri bilimsel çalışmalarda beni yönlendiren, vizyonumu genişleten Dr. S. Faegheh YEGANLİ'ya en içten ve samimi teşekkürlerimi sunarım. Ayrıca, küçük yaşlardan itibaren, bilimsel çalışmaların ülkemiz ve coğrafyamızın gelişimi için taşıdığı önemi kavrayarak, bana disiplin ve vizyon kazandıran, her koşulda maddi ve manevi olarak destekleyen aileme teşekkür ederim.

Muhammed Faruk Şahin

İŞBİRLİKÇİ ZEKA YAKLAŞIMIYLA ÇOK GÖREVLİ ÖĞRENMEDE DİNAMİK BÖLME NOKTASI HESAPLAMA

Muhammed Faruk Şahin

ÖZET

Derin Sinir Ağları'nın (DNN'ler) Nesnelerin İnterneti (IoT) ortamlarında, özellikle çoklu görevli robotik ve sürü sistemleri için dağıtımında donanım sınırlamaları, bant genişliği kısıtlamaları, iletim gecikmeleri ve görüntü paketi kaybı olmak üzere çeşitli zorlukları bulunmaktadır. Bu tez çalışmasında, İşbirlikçi Zeka (CI) paradigması altında Çoklu Görev Öğrenme (MTL) tabanlı hesaplama görevlerini uç cihazlar ve bulut arasında dinamik olarak dağıtan bir DSPCI-MTL çerçevesi önerilmektedir. Önerilen çerçeve, gerçek zamanlı bant genişliği ve veri hacmi temelinde DNN katmanları için bölme noktalarını segmentasyon, sınıflandırma, derinlik tahmini görevleri için optimize ederken özellik haritası benzerliği yoluyla kayıp görüntü paketlerini yeniden oluşturmak için bir Otomatik Kodlayıcı (AE) mimarisini entegre etmektedir. Deneysel sonuçlar, geleneksel bulut tabanlı yöntemlere kıyasla işleme süresinde %38'lik bir azalma ve dinamik bölme noktası seçimiyle %61'lik bir kazanç olduğunu göstermektedir. AE tabanlı yeniden yapılandırma yöntemi, karmaşık ve uzun mesafeden çekilen görüntüleri iletim senaryolarında veri bütünlüğünü önemli ölçüde iyileştirerek kaynak kısıtlı IoT uygulamaları için gelişmiş sistem performansı sunmaktadır. Ek olarak, kaynak sınırlı IoT cihazlarında DNN'leri dağıtmak için hesaplama yükü ve bant genişliği verimliliğinin dengelenmesini gerektirmektedir. Bu sebeple, Gri Kurt Optimizasyonu (GWO), Parçacık Sürüsü Optimizasyonu (PSO), Yapay Arı Kolonisi (ABC) ve Yapay Tavşan Optimizasyonu'ndan (ARO) yararlanarak uç ve bulut arasındaki optimum DNN bölme noktalarını uyarlamalı bir şekilde belirlemek için Meta-Sezgisel Algoritmalarla Dinamik Bölme Noktası Belirlenmesi amacıyla DSPCI-MH tanıtılmaktadır. Statik yaklaşımların aksine, DSPCI-MH ağ koşullarına ve hesaplama

taleplerine dinamik olarak uyum sağlayarak geleneksel yöntemlere kıyasla %99,86 daha hızlı çıkarım, %99,85 daha düşük enerji tüketimi ve %99,98 iyileştirilmiş bellek kullanımı elde etmektedir. Çerçevenin gerçek zamanlı uyarlanabilirliği, dağıtılmış DNN çıkarımı için ölçeklenebilir bir çözüm oluşturarak yapay zeka odaklı IoT sistemlerindeki kritik zorlukları ele almaktadır. Sonuçlar, metasezgisel optimizasyonu dinamik ortamlarda uç-bulut iş birliğini geliştirmek için dönüştürücü bir strateji olarak doğrulamaktadır.

Anahtar Kelimeler: İşbirlikçi Zeka, Derin Sinir Ağları, Nesnelerin İnterneti, Metasezgisel, Dinamik Bölme Noktası Hesaplama, Veri Yeniden Yapılandırma

DYNAMIC SPLIT POINT COMPUTING IN MULTI-TASK LEARNING IMPLEMENTATION WITH COLLABORATIVE INTELLIGENCE

Muhammed Faruk Şahin

ABSTRACT

Deep Neural Networks (DNNs) face several challenges in deployment within Internet of Things (IoT) environments, particularly for multi-task robotic and swarm systems. These challenges include hardware limitations, bandwidth constraints, transmission delays, and image packet loss. This thesis proposes a DSPCI-MTL framework under the Collaborative Intelligence (CI) paradigm, which dynamically distributes Multi-Task Learning (MTL)-based computational tasks between edge devices and the cloud. The proposed framework optimizes partitioning points for DNN layers based on real-time bandwidth and data volume while integrating an Autoencoder (AE) architecture to reconstruct lost image packets through feature map similarity for segmentation, classification, and depth estimation tasks. Experimental results demonstrate a 38% reduction in processing time and a 61% improvement in dynamic partition point selection compared to traditional cloud-based methods. The AE-based reconstruction method significantly enhances data integrity in transmission scenarios involving complex and long-range images, providing improved system performance for resource-constrained IoT applications. Additionally, deploying DNNs on resource-limited IoT devices requires balancing computational load and bandwidth efficiency. To achieve Dynamic Partition Point Determination with Metaheuristic Algorithms, the DSPCI-MH approach is introduced. This method adaptively determines optimal DNN partition points between the edge and the cloud by leveraging Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Artificial Rabbit Optimization (ARO). Unlike static approaches, DSPCI-MH dynamically adapts to network conditions and

computational demands, achieving 99.86% faster inference, 99.85% lower energy consumption, and 99.98% improved memory utilization compared to conventional methods. The real-time adaptability of this framework provides a scalable solution for distributed DNN inference, addressing critical challenges in AI-driven IoT systems. The results validate metaheuristic optimization as a transformative strategy for enhancing edge-cloud collaboration in dynamic environments.

Keywords: Collaborative Intelligence, Deep Neural Network, IoT, Meta-Heuristics, Dynamic Split Point Determination, Data Reconstruction

ÖN SÖZ

Yapay Zekâ ve Nesnelerin İnterneti (IoT) teknolojilerinin entegrasyonu, akıllı sistemlerin verimliliğini artırırken hesaplama gücü, bant genişliği, veri iletim kayıpları ve çeşitli sorunları da beraberinde getirmektedir. IoT ortamlarında Derin Sinir Ağları (DNN) tabanlı uygulamaların verimli bir şekilde çalışabilmesi için görev dağılımı, işlem süreleri ve veri bütünlüğünün optimize edilmesi kritik bir öneme sahip olmaktadır. Bu bağlamda gerçekleştirilen tez çalışmasında, yaşanan temel sorunlara yönelik iki yenilikçi çözüm önerisi sunulmaktadır. İlk olarak, İşbirlikçi Zekâ (Collaborative Intelligence-CI) stratejileri ve Çoklu Görev Öğrenimi (Multi-Task Learning-MTL) yaklaşımı kullanılarak dağıtık DNN çıkarımı optimize edilmektedir. Bu kapsamda, uç ve bulut arasında dinamik hesaplama dağılımı sağlanmakta ve DNN'nin bölme noktaları, ağ koşulları ve veri büyüklüğüne göre dinamik olarak belirlenmektedir. Ayrıca, iletim sırasında meydana gelen veri kayıplarını gidermek amacıyla Otomatik Kodlayıcı (Auto Encoder-AE) mimarisi kullanılmaktadır. İkinci olarak, bölme noktalarının belirlenmesi için Meta-Sezgisel Optimizasyon yöntemleri kullanılarak sistemin adaptif yetenekleri artırılmaktadır. Bu bağlamda geliştirilen Dinamik Bölme Noktasını Meta-Sezgisel Algoritmalarla belirleme (DSPCI-MH) çerçevesinde, Gri Kurt Optimizasyonu (GWO), Parçacık Sürü Optimizasyonu (PSO), Yapay Arı Kolonisi (ABC) ve Yapay Tavşan Optimizasyonu (ARO) kullanılarak hesaplama yükü verimli şekilde dağıtılmaktadır. Böylece, önerilen yaklaşımın geleneksel statik yöntemlere kıyasla çok daha dinamik ve uyarlanabilir bir çözüm sunduğunu kanıtlamaktadır. Tez çalışması sırasında gerek teorik altyapının oluşturulması gerekse deneysel analizlerin gerçekleştirilmesi süreci önemli zorluklar içermektedir. Geniş ölçekli veri setleriyle çalışmanın getirdiği yüksek işlem yükü, IoT ortamındaki değişken ağ koşullarına adaptasyon sağlama gerekliliği ve algoritmaların gerçek zamanlı optimizasyonu araştırma sürecinin en büyük teknik zorluklarını oluşturmaktadır. Ancak, bu zorluklar bilimsel problem çözme yaklaşımı ve sistematik deney tasarımıyla aşılarak geliştirilen çözümler

gelecekteki araştırmalara ışık tutabilecek nitelikte olmaktadır. Bu tez çalışmasının, yapay zeka ve IoT'nin kesişim noktasında yapılan araştırmalara önemli bir katkı sağlaması ve gelecekte bu alanda çalışacak bilim insanlarına rehberlik etmesi en büyük temenniyi oluşturmaktadır.

Mayıs, 2025 Muhammed Faruk Şahin

İÇİNDEKİLER

SEMBOLLER

T^{de}	: Derinlik Tahmini Görevi
T^{s}	: Görüntü Segmentasyonu Görevi
T^{cl}	: Görüntü Sınıflandırma Görevi
T	: Görevler
k	: Her Bir Sınıflandırıcı
i	: Görev İndisi
m	: Öğrenilen Görev Sayısı
n	: Eğitilen Örnek Sayısı
j	: Örnek İndisi
x_j	: Veri
y_j	: Etiket Değeri
e	: Derinlik Tahmini
R	: Satır Sayısı
C	: Sütun Sayısı
ch	: Kanal Sayısı
x_j	: Giriş Görüntüsü
f_1	: Derin Sinir Ağının İlk Katmanı
f_L	: Derin Sinir Ağının Son Katmanı
c_L	: Katmanın Sıkıştırma Oranı
h	: Uç Tarafın Son Katmanın Çıktısı
$V(h_L)$	: L'inci Katmanın Nicemleme (ing: Quantization) Çıktısı
IT	: Çıkarım Süresi
$IT^{f}_{1,z}$	: İlk Katmandan Z'inci Katmana Çıkarım Süresi
$IT^{b}_{z+1,k}$	: Uç Cihazdan Bulut Cihaza İşlem Süresi
$IT^{b}_{k+1,L}$	: Bulut Cihazda Çıkarım Süresi
$c_z IT_{FO}$	: Bulut Cihazda Sıkıştırma Oranı
IT_{FO}	: Bölme Noktası Olmaması Durumunda Çıkarım Süresi
Dc_p	: Sıkıştırma ve İletim İçin Hazırlanan Veriler
r	: Bant Genişliği
$\frac{Dc_p}{r}$	: Z'inci Katmanın Sıkıştırma Oranı
$IT^{b}_{z+1,L}$	: z+1'inci Katmandan Son Katmana Çıkarım Süresi
S_{opt}	: Optimum Bölme Noktası
w_1 ve w_2	: Sırasıyla Hafıza ve Enerji Tüketimi Katsayısı
M_z	: Hafıza Kullanımı

$\boldsymbol{E_z}$: Enerji Tüketimi
$\boldsymbol{X_{leader}}$: Alfa Kurt
$\boldsymbol{X_t}$: Güncel Çözümün Pozisyonu
$\boldsymbol{D}$: En İyi Çözüme Uzaklık
$\boldsymbol{C}$: En İyi Çözüme Olan Uzaklığı Ölçeklendiren Katsayı
$\boldsymbol{r_1}$ ve $\boldsymbol{r_2}$: 0-1 Aralığında Rastgele Değer Üreten Katsayı
$\boldsymbol{X_1}$, $\boldsymbol{X_2}$ ve $\boldsymbol{X_3}$: Kurtların Güncel Pozisyonları
$\boldsymbol{A}$: GWO Algoritmasında İterasyona Bağlı Arama Katsayısı
$\boldsymbol{a}$: Lineer Olarak İkiden Sıfıra Düşen İterasyon İşlemi
$\boldsymbol{X_i}$: Konum Vektörü
$\boldsymbol{V_i}$: Hız Vektörü
$\boldsymbol{P_i^{best}}$: Sürüdeki Parçanın En İyi Konumu
$\boldsymbol{G_g^{best}}$: Küresel En iyi Konum
$\boldsymbol{X_i}$: Güncel Çözüm
$\boldsymbol{X_j}$: Rasgele Seçilen Komşu Çözüm
ϕ : Çeşitliliği Arttırmak İçin [-1,1] Aralığında Rastgele Seçilen Katsayı
$f(X_i)$: Çözümün Uygunluk Değeri
$F(t)$: Zamana Bağlı Optimizasyon Faktörü
δ : Keşif-Sömürü Faktörü

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

KISALTMALAR

DNN	Derin Sinir Ağı / Deep Neural Network
IoT	Nesnelerin İnterneti / Internet of Things
MTL	Çok Görevli Öğrenme / Multi-task Learning
CI.	İşbirlikçi Zeka / Collaborative Intelligence
AE.	Oto kodlayıcı / Auto Encoder
AI	Yapay Zeka / Artificial Intelligence
E2C	Uç-Bulut / Edge-to-Cloud
E2E	Uç-Uç / Edge-to-Edge
RMSE	Kök Ortalama Kare / Root Mean Square
IoU	Kesişim ve Birlik Oranı / Intersection Over Union
mIoU	Ortalama Kesişim ve Birlik Oranı / Mean IoU
SSIM	Yapısal Benzerlik Endeksi Ölçüsü
PSNR	Tepe Sinyal-Gürültü Oranı / Peak Signal-To-Noise Ratio
MAE	Ortalama Mutlak Hata / Mean Absolute Error
IT	Çıkarım Süresi / Inference Time
MH	Metasezgisel / Meta-Heuristics
GWO	Gri Kurt Optimizasyonu / Grey Wolf Optimization
PSO	Parçacık Sürü Optimizasyonu / Particle Swarm Optimization
ABC	Yapay Arı Optimizasyonu / Artificial Bee Colony
ARO	Yapay Tavşan Optimizasyonu / Artificial Rabbit Optimization
GPU	Grafik İşlem Birimi/ Graphics Processing Unit
CPU	Merkezi İşlem Birimi/ Central Process Unit

GİRİŞ

Nesnelerin İnterneti (ing: Internet of Things-IoT) çerçevelerinin hızlı gelişimi yapay zeka (ing: Artificial Intelligence-AI) alanındaki ilerlemelerle birlikte IoT sistemleri içindeki AI yeteneklerinin entegrasyonunu önemli ölçüde artırarak AI destekli IoT uygulamalarının daha geniş bir şekilde konuşlandırılmasına yol açmaktadır (Rodriguez-Conde vd., 2023; Seyyedabbasi vd., 2023). Özellikle, Derin Sinir Ağları (ing: Deep Neural Network-DNN), yüksek öngörü doğrulukları sayesinde, otonom araçlar, çok görevli robotlar ve sürü sistemlerinde bulunan karmaşık görüntü işleme gerektiren uygulamalarda giderek daha fazla tercih edilmektedir (Abd El Bar vd., 2015; Chakraa vd., 2023; X. Wang vd., 2023; Yatbaz vd., 2024). Bununla birlikte, DNN'lerin kaynak kısıtlı IoT cihazlarında uygulanması durumunda bant genişliği sınırlamaları ve veri iletim sürelerindeki artış olmak üzere çeşitli zorlukları da beraberinde getirmektedir. Bu durum, enerji tüketimini artırırken operasyonel verimliliği azaltmaktadır (Ahmad vd., 2021; Howard vd., 2017; Kang vd., 2017; Li vd., 2020; Singh & Gill, 2023). Söz konusu sorunlar, özellikle görüntü verisi üreten homojen ve heterojen IoT cihazlarında giderek daha belirgin hale gelmektedir. Gerçek dünya uygulamalarında, düşük gecikme süresinin kritik olduğu senaryolarda, bu kaynak sınırlamalarına sahip cihazlardan gelen verilerin hızlı ve verimli bir şekilde işlenmesi büyük önem taşımaktadır. Bu sebeple, verimliliği arttırmak amacıyla, DNN'i uç (ing: edge) ve bulut (ing: cloud) cihazlarına bölerek işleyen İşbirlikçi Zeka (ing: Collaborative Intelligence-CI) kavramı yenilikçi bir paradigma olarak öne çıkmaktadır (Cohen vd., 2021). CI çerçevesinde, uç cihazlar ön işleme ve özellik hesaplama görevlerini üstlenirken uç cihazın çıktısı olan ara özellikler buluta iletilerek çıkarım işlemi tamamlanmaktadır (Shiranthika vd., 2023). CI modeli, Uçtan-Buluta (ing: Edge-to-Cloud - E2C) ve Uçtan-Uca (ing: Edge-to-Edge - E2E) yapılandırmalarını destekleyecek şekilde esnek bir mimariye sahip olmaktadır (Bajic vd., 2021). Böylece CI mimarisi, hesaplama görevlerini uç cihazlar ve bulut arasında dağıtarak bireysel cihazların tespit, hesaplama ve depolama

kapasiteleriyle ilgili kısıtlamalarını iyileştirmektedir. Özellikle uç cihazlarda gerçekleştirilen veri işleme, girdi verilerinden türetilen özellik haritalarına dayandığından, kullanıcı verilerinin gizliliğinin korunmasına da katkıda bulunmaktadır. Buna ek olarak, uç ve bulut cihazları arasındaki bu işbirlikçi yaklaşım yalnızca enerji tüketimini minimize etmekle kalmayıp, modelin genel performansını da önemli ölçüde artırmaktadır (Danba vd., 2022). Ancak tüm bu avantajlara rağmen, IoT sistemlerinin doğası gereği çoğunlukla birden fazla çıktıya ihtiyaç duyması genellikle tekil görevler için optimize edilen CI tabanlı mimariler açısından önemli bir zorluk teşkil etmektedir. Bu kısıt, birden fazla görevi eşzamanlı olarak işleyebilen ve böylece modelin genel performansını ve veri verimliliğini artırırken aşırı öğrenme (ing: overfitting) riskini azaltan Çok Görevli Öğrenme (ing: Multi-Task Learning-MTL) yaklaşımlarının geliştirilmesini teşvik edilmektedir (Azadi vd., 2024; Crawshaw, 2020; Y. Zhang & Yang, 2022; Zhao vd., 2023). MTL, CI ile entegre edildiğinde bulut tabanlı sistemlerde çok görevli işlemleri etkili bir şekilde destekleyerek sürü tabanlı ve çok görevli sistemlerin yeteneklerini artırmaktadır (Alvar & Bajic, 2019). Bununla birlikte, hibrit CI-MTL modelleri, ağ durumunun dinamik olarak değişmesi ve veri hacmindeki dalgalanmalar gibi faktörler nedeniyle zorluklarla karşılaşmaktadır. Bu durum hem uç hem de bulut cihazlarında verimliliği etkileyerek veri kaybını artırmakta ve tahmin yanıtlarında gecikmelere neden olmaktadır. Ayrıca, CI çerçevesinde kritik bir bileşen ise DNN içinde optimal bölme noktasının belirlenmesi olmaktadır. DNN'in, hem uç hem de bulut cihazları tarafından verimli bir şekilde işlenmesini sağlayacak şekilde uygun bir noktada bölünmesi, çıkarım süresini, enerji tüketimini ve bellek performansını iyileştirmektedir (Alvar & Bajic, 2020). Ancak, değişken ağ koşulları ve potansiyel iş yükü değişikliklerine uyum sağlamak için doğru bölme noktasının belirlenmesi kritik bir öneme sahip olmaktadır (Bakhtiarnia vd., 2023). Bu stratejik operasyon, yalnızca hesaplama kaynaklarını optimize etmekle kalmayıp aynı zamanda sistemin tepki süresini, kaynak kullanımını ve genel verimliliğini de artırmaktadır (Karjee vd., 2021). Bu bağlamda, DNN'lerde bölme noktasının belirlenmesine yönelik rastgele ve dinamik seçim olmak üzere iki farklı senaryo bulunmaktadır (Matsubara vd., 2023). Rastgele seçim yöntemi belirli avantajlar sağlasa da en verimli kazançlara ulaşma konusunda yetersiz kalmaktadır. Alternatif senaryoda ise ağ koşulları, DNN

mimarisi, veri iletim hacmi ve çeşitli parametrelere dayalı olarak, matematiksel veya sezgisel (ing: heuristic) yaklaşımlar kullanılarak bölme noktasının dinamik olarak belirlenmesi hedeflenmektedir (Duan & Wu, 2021). Ancak bu durumda, gerçek zamanlı uygulamalarda hesaplama yükü, çıkarım gecikmesi, bellek kullanımı ve enerji maliyetleri ve çeşitli zorluklar ortaya çıkmaktadır (Jahier Pagliari vd., 2020; Yao vd., 2020). Bu sebeple, rastgele seçim yöntemine kıyasla daha yüksek verimlilik sağlaması nedeniyle bölme katmanının dinamik olarak belirlenmesi bu zorlukların azaltılması açısından kritik bir öneme sahip olmaktadır.

Bilinen literatürde, bölme noktalarının rastgele belirlenmesi ve özellik haritalarının buluta iletimi sırasında oluşan veri kaybı olmak üzere iki temel zorluğa dikkat çekmektedir. Rastgele belirlenen bölme noktaları, DNN tahmin süreçleri için net bir yönlendirme sağlamaması sebebiyle model performansını optimize etmede yetersiz kalmaktadır. Öte yandan, veri kaybı, DNN'in performansını önemli ölçüde olumsuz etkileyebilmektedir. Gerçek zamanlı koşullara dayalı dinamik bölme durumunda ise bölme noktasının belirlenmesi enerji tüketimi, hesaplama maliyeti ve bellek kullanımı açısından çeşitli zorluklar ortaya çıkarmaktadır. Bu sorunları ele almak amacıyla, ağ cihazlarındaki hesaplama yükünü en aza indirmeyi hedefleyen yenilikçi ve dinamik bir DNN bölme yöntemi önermektedir. Ayrıca, veri kaybını gidermek için, iletim sırasında kanal içi ilişkileri analiz eden veri yeniden yapılandırma yöntemlerini kullanan bir Otomatik Kodlayıcı (ing: Auto Encoder-AE) tabanlı yeniden yapılandırma mimarisi önerilmektedir. Önerilen veri yeniden yapılandırma yöntemi, ilgili çalışmalara kıyasla daha karmaşık görüntüler üzerinde test edilmiştir. Bununla birlikte, belirli ağ koşullarına ve veri hacimlerine uyum sağlayacak şekilde tasarlanan yapı MTL ile entegre edilerek gelişmiş bir hibrit CI çerçevesi geliştirilmektedir. MTL ve CI'nin birleşimi, mevcut araştırmalardaki bir boşluğu doldurarak, önceki çalışmaların ele almadığı zorlukları aynı anda çözmeyi amaçlamakta ve bu alanda önemli bir ilerleme sunmaktadır. Öte yandan, karmaşık ve gerçek zamanlı DNN uygulamalarında yerel minimumlara duyarlılık, hiperparametre optimizasyonundaki zorluklar, enerji ve hesaplama maliyetleri kısıtlamaların üstesinden gelmek amacıyla Meta-Sezgisel (ing: Metaheuristic-MH) algoritmalar etkili bir yaklaşım sunmaktadır (Cuong-Le vd., 2022; Lee vd., 2020; Thakkar & Lohiya, 2023). Sonuç olarak, DNN ve görüntü tabanlı uygulamalarda performansı

artırmak için MH algoritmalarının kullanımı giderek yaygınlaşmaktadır (Akay vd., 2022; Darwish vd., 2020). MH algoritmaları, optimizasyon problemlerine sağlam ve esnek çözümler sunarak DNN uygulamalarında dönüştürücü bir rol oynamaktadır (Riaz vd., 2022; Tomar vd., 2024). Bu nedenle, bu çalışma, bölme noktasının dinamik olarak belirlenmesi için MH algoritmalarının da kullanımını önermektedir. Önerilen yöntem, gerçek zamanlı uygulamalar için daha kapsamlı ve verimli bir bölme noktası seçimi sağlamaktadır. MH algoritmalarının CI ile entegrasyonu, mevcut araştırmalardaki bir boşluğu doldurarak, daha önce aynı anda ele alınmamış olan zorlukları kapsamlı bir şekilde ele almaktadır. Bu katkı, alanında önemli bir ilerlemeyi temsil etmektedir. Bu çalışmanın temel katkıları aşağıdaki şekilde özetlenebilir:

- Geleneksel CI sistemlerinin kısıtlarını aşarak verimli çok görevli işlemeyi mümkün kılan, MTL tabanlı geliştirilmiş bir hibrit model önerilmektedir.
- CI-MTL hibrit çerçevesi içerisinde, DNN mimarilerinde bölme noktasının dinamik olarak optimize edilmesini sağlayan ve ağ ile yük değişikliklerine gerçek zamanlı olarak uyum sağlayan bir yöntem geliştirilmiştir.
- Bulutta, özellik haritalarından elde edilen benzerlik metriklerini kullanarak kaybolan veri paketlerini yeniden yapılandırmak için özel olarak tasarlanmış yenilikçi bir AE mimarisi önerilmektedir. Bu yaklaşım, özellikle ayrıntılı ve uzak mesafedeki görüntülerde veri bütünlüğünü artırmaya yöneliktir.
- Optimize edilmiş veri işleme ve stratejik görev dağılımı ile sistemin tepki süresi iyileştirilmiş ve veri güvenliği sağlanmıştır.
- Uç ve Bulut cihazları arasında DNN modellerinin bölünmesini optimize etmek amacıyla, değişken ağ koşulları, hesaplama kapasitesi, bellek kullanımı ve enerji tüketimi gibi faktörleri dikkate alan dinamik ve uyarlanabilir bir MH tabanlı yöntem geliştirilmiştir.
- Önerilen yaklaşım, hesaplama yükünü dengeleyerek, bellek tüketimini ve enerji maliyetlerini optimize ederek ve DNN süreçlerindeki çıkarım gecikmesini azaltarak sistemin genel verimliliğini artırmayı hedeflemektedir.

- Mevcut rastgele veya statik bölme yöntemlerinden farklı olarak, MH algoritmalarından yararlanılarak DNN'nin farklı katmanları için en uygun bölme noktalarını belirleyen uyarlanabilir bir bölme stratejisi sunulmaktadır.
- Önerilen MH tabanlı bölme yöntemi, rastgele ve dinamik bölme yaklaşımlarına karşı değerlendirilerek ağ koşullarına uyarlanabilirliği ve sistem verimliliğindeki iyileştirmeleri açısından üstün performans sergilediği gösterilmektedir.

Bu tezin geri kalanı şu şekilde düzenlenmiştir. Bölüm 1 literatürdeki ilgili çalışmaları özetlemektedir. Bölüm 2 önerilen DSPCI-MTL yöntemini açıklayarak deneysel sonuçlarını sunmaktadır. Bölüm 3 önerilen DSPCI-MH yöntemini açıklayarak deneysel sonuçlarını sunmaktadır. Bölüm 4 çalışmanın sınırlamalarını araştırmakta ve bulguların daha geniş bir tartışmasını sunmaktadır. Son olarak, sonuç kısmı makaleyi sonlandırmakta, temel sonuçları ve gelecekteki araştırmalar için olası yönleri vurgulamaktadır.

BİRİNCİ BÖLÜM

1. TEMEL KAVRAMLAR VE LİTERATÜR TARAMASI

Bu kısımda, tez çalışmasının temel odağını oluşturan görüntü verilerine uygulanan CI ve DNN katman bölümlemesi ile ilgili bilinen temel kavramlar ve literatürdeki çalışmalar incelenmektedir. Bu çalışmalar, önerilen DSPCI-MTL ve DSPCI-MH yaklaşımları için ilgili metodolojileri temsil etmektedir.

Günümüzde derin öğrenmenin uç ve bulut sistemleri arasında dengeli ve verimli bir şekilde uygulanabilmesi için model bölme (ing:model partitioning), görev aktarma (ing:task offloading) ve çeşitli teknikler geliştirilmektedir. Bu bağlamda işbirlikçi zeka kavramı, uç cihazlar ve bulut sistemleri arasında iş yükünün dinamik ve verimli şekilde paylaşılmasını ifade etmektedir. Hesaplama kaynaklarının sınırlı olduğu uç cihazlarda yalnızca belirli katmanların çalıştırılması kalan işlemlerin ise daha güçlü donanıma sahip bulut sistemine aktarılması sayesinde hem enerji tüketimi azalmakta hem de yanıt süresi iyileştirilmektedir. Bu türdeki dağıtık yapılarda modelin farklı görevleri aynı anda öğrenebilmesi amacıyla çok görevli öğrenme yaklaşımı kullanılmaktadır. Çok görevli öğrenme, ilişkili görevlerin aynı model altında ortak temsiller üzerinden öğrenilmesini sağlayarak genel performansı artırmakta ve parametre paylaşımı sayesinde aşırı öğrenmenin önüne geçmektedir. Bu yöntem, özellikle sınırlı veri veya donanım koşullarında daha etkili hale gelmektedir. Bu süreçte modelin, uç ve bulut cihazda çalışacak katmanlarının belirlenmesi için ayrım noktası belirleme sorunu DNN bölme noktası belirleme olarak ifade edilmektedir. Bu bölünme noktasının doğru seçilmesi, iletişim yükü, hesaplama maliyeti ve model doğruluğu açısından kritik öneme sahip olmaktadır. Bölme noktası seçiminde önceden belirlenen sabit stratejiler yerine, veri yapısına ve sistem durumuna göre dinamik yaklaşımlar daha esnek ve etkili çözümler sunmaktadır. Öte yandan, uçtan buluta veri aktarımı yapılırken ağ üzerinde yük oluşturmamak için verinin anlamlı ve sıkıştırılmış bir temsil halinde iletilmesi gerekmektedir. Bu durum, ağ üzerinde paket kayıpları oluşması sebebiyle veri

yeniden yapılandırma tekniklerine ihtiyacı oluşturmaktadır. Özellikle derin öğrenme tabanlı modellerde, ara katman çıkışlarının daha az yer kaplayacak şekilde dönüştürülmesi, bant genişliği ihtiyacını azaltırken bilgi kaybını da en aza indirmeye yardımcı olmaktadır. Bu bölüm, önerilen DSPCI-MTL yönteminin temel odağını oluşturan görüntü verilerinde kullanılan CI ile ilgili önceki çalışmaları incelemektedir. Bu çalışmalar, her biri farklı bakış açılarını ve metodolojileri temsil eden dört sınıfa ayrılmaktadır. Bu çalışmaların özellikleri, avantajları ve sınırlamaları Tablo 1'de özetlenmektedir.

1.1. İŞBİRLİKÇİ ZEKA VE ÇOKLU GÖREV ÖĞRENME

IoT sistemlerindeki cihazların aynı anda birden fazla görevi ele alma zorunluluğu sebebiyle MTL'i DNN mimarilerine dahil etmek giderek daha da önemli hale gelmektedir. MTL, sistemlerin farklı görevleri aynı anda ele alma yeteneğini geliştirerek genel sistem verimliliğini ve farklı operasyonel ortamlarda veri işlemeyi iyileştirmektedir. MTL'in CI çerçeveleri içinde uygulanması, stratejik veri yönetimi yoluyla kullanıcı gizliliğini korurken hesaplama yükünü önemli ölçüde azaltmaktadır. Bu bağlamda, Alvar ve ark. (Alvar & Bajic, 2019), MTL'i CI çerçeveleriyle entegre ederek bir DNN modelini Cityscapes veri kümesi (Cordts vd., 2016) üzerinde eğitmektedir. Bu yaklaşım, modelin aynı anda birden fazla görevi ele almasına olanak sağlamaktadır. Ancak, bu çalışma DNN bölme noktalarının dinamik belirlenmesinde bir boşluğu vurgulamaktadır. MTL'in CI çerçevelerine uygulanmasını daha fazla inceleyen başka bir çalışma, E2C yapılandırmalarının tam CI modeli olmadan araç davranışını izleme operasyonel ihtiyaçlarına nasıl uyum sağlayabileceğini göstermektedir. Bu uyarlama, hesaplama maliyetini azaltırken kullanıcı gizliliğini korumak için kritik olmaktadır (Alvar vd., 2022). MTL'i CI ile entegre etmenin avantajlarına rağmen, DNN mimarileri içinde bölme noktalarının dinamik hesaplanmasında önemli bir zorluk devam etmektedir. Bu sınırlama, özellikle dalgalanan ağ koşullarında, genellikle optimum olmayan performansa yol açmaktadır. Bu sorunu iyileştirmek için, bölme noktası belirlemesi yapılmaktadır.

1.2. BÖLÜNME Noktası Belirleme

Bu bölümde, önerilen DSPCI-MTL ve DSPCI-MH yöntemleri için DNN katman bölümlemesi ile ilgili literatürdeki bilinen çalışmalar incelenmektedir. Bu çalışmalar, her biri farklı bakış açılarını ve metodolojileri temsil eden rastgele veya kodlama tabanlı yöntemler olmak üzere matematiksel tabanlı yöntemler ve sezgisel olmak üzere iki sınıfa ayrılmıştır. Bu çalışmaların özellikleri, avantajları ve sınırlamaları Tablo 2'de özetlenmektedir. Tartışıldığı üzere, CI bağlamında, bir DNN'deki optimum bölme noktasını belirlemek, uç cihazlar ve bulut arasındaki iş yükünü dengelemek için kritik öneme sahip olmaktadır. Bu stratejik segmentasyon, verimli işlemeyi garanti ederken çıkarım sürelerini iyileştirir ve veri hacmi (ing: batch size) ve ağ bant genişliğine (ing: bandwidth) göre dinamik olarak ayarlanarak kullanıcı verilerini güvence altına almaktadır. Rastgele belirlenen bölme noktası performansı düşürebilir ve değişen ağ koşulları ve veri paketi modeli nedeniyle veri kaybına yol açmaktadır (Z. Zhang vd., 2021). Bu nedenle, cihazların aşırı yüklenmesini önlemek ve verimli çalışmayı sağlamak için bölme noktasının veri hacmi ve ağ koşullarına göre dinamik olarak belirlemek gerekli olmaktadır.

1.2.1. Matematik Tabanlı Yöntemler

DNN katmanlarının bölünmesinde, ilk olarak mobil ve bulut ortamlarında hesaplama döngülerini verimli bir şekilde kullanmak için DNN algoritmalarının karakteristik özellikleri analiz edilmekte ve bu durum Neurosurgeon yönteminin önerilmesine yol açmaktadır (Kang vd., 2017). Önerilen yöntem, DNN hesaplamalarını mobil cihazlar ve veri merkezleri arasında katman düzeyinde otomatik olarak bölen bir zamanlayıcı işlevi görmektedir. DNN katmanlarının bölümlenmesini verilere ve hesaplama değişkenliğine göre uyarlayarak gecikme ve enerji verimliliği avantajları elde ederek gecikmeyi ortalama 3,1 kat azaltmaktadır. Ancak sınıflandırıcılara sınırlı uygulanabilirliği nedeniyle, JointDNN yaklaşımı mobil cihazlar ve bulut arasında işbirlikçi hesaplamayı kolaylaştırmak için tanıtılmaktadır (Eshratifar vd., 2021). Önerilen yöntem, uygulama alanını genişleterek ortalama %4,6 gecikme iyileştirmesi sağlamaktadır. Ek olarak, tüm bu çalışmalar yalnızca yerel veya bulut bilişimine güvenmenin çıkarım süresi ve enerji tüketimi açısından optimum bir çözüm sağlamadığını göstermektedir. Sınırlı

hesaplama kaynakları nedeniyle, mobil cihazların hesaplama açısından yoğun görevleri yürütmesini sağlamak için Edgent önerilmektedir (Li vd., 2020). Önerilen yaklaşım, erken çıkış noktalarını DNN ölçeklemesiyle bütünleştirerek ara katman çıkışlarının çıkarım süresini artırmasına olanak tanımaktadır. Buna karşılık, orijinal ağa erken çıkış noktaları ekleyen ve onu iki bölüme ayıran ADDA yöntemi önerilmektedir (H. Wang vd., 2019). Önerilen yaklaşım, DNN karmaşıklığından kaynaklanan gecikmeyi %6,6 oranında azaltmaktadır. Ancak, her iki yöntem de optimum bölümlemeden yoksun olduğundan, Karma Tamsayı Doğrusal Programlama (MILP) alternatif bir çözüm olarak önerilmekte ve %69,5 gecikme iyileştirmesi sağlanmaktadır (Gao vd., 2019).

Ancak, CI bağlamında, bir DNN'deki optimum bölme noktasını belirlemek, uç cihazlar ile bulut arasındaki iş yükünü dengelemek için kritik öneme sahip olmaktadır. Bu nedenle, veri hacmi ve ağ bant genişliğine göre dinamik olarak ayarlanması gerekmektedir (Capogrosso vd., 2023). Ancak, CI bağlamında, bir DNN'deki optimum bölme noktasını belirlemek, uç cihazlar ile bulut arasındaki iş yükünü dengelemek için kritik öneme sahip olmaktadır. Bu nedenle, veri hacmi ve ağ bant genişliğine göre dinamik olarak ayarlanması gerekmektedir (Bakhtiarnia vd., 2023). Bu yöntem, iletişim kanalının durumuna göre optimum bölme noktasını dinamik olarak seçmek için modern DNN mimarilerindeki doğal darboğazlardan yararlanarak yeniden eğitim ve hiperparametre optimizasyonundan kaçınırken nihai doğruluk üzerinde olumsuz bir etki olmamasını sağlamaktadır. Benzer şekilde, Karjee ve diğerleri (Karjee, Naik S, vd., 2022), IoT sistemlerindeki mevcut hesaplama gecikmesi ve ağ koşullarına göre E2E ve E2C arasında DNN katmanlarını optimum şekilde dağıtmak için dinamik bir bölümleme mekanizması sunarak hesaplama yükünü azaltan bir Dinamik Bölme Hesaplama tekniği önermektedir. Önerilen yöntem, mevcut ağ bant genişliğine dayalı olarak DNN katmanları için en uygun bölme noktasını belirlemeye odaklanan Genişletilmiş Dinamik Bölme Hesaplaması (E-DSC) (Karjee, Anand, vd., 2022) adlı genişletilmiş bir versiyonda uyarlanarak uygulanmaktadır. Ancak, dinamik ağ koşullarında matematiksel yöntemlerin hesaplama maliyeti ve uyarlanabilirlik sınırlamaları, bölme katmanını belirlemek için daha pratik ve uyarlanabilir çözümler sunan sezgisel yaklaşımların artan önemini vurgulamaktadır.

1.2.2. Sezgisel Tabanlı Yöntemler

Son yıllarda sezgisel yöntemler, DNN'lerdeki bölünme katmanı belirleme sorununa pratik ve esnek çözümler sağlamak için önemli ilgi görmektedir (Xu vd., 2023). Bu yaklaşımlar, matematiksel modellerin karmaşıklığını ve katılığını aşarak dinamik ağ koşullarına, veri akışına ve uygulama senaryolarına daha hızlı adaptasyona olanak tanımaktadır. Bu bölüm, bölünme katmanı seçimi için kullanılan çeşitli sezgisel yöntemleri ve bunların performans üzerindeki etkilerini incelemektedir.

Büyük ölçekli DNN'ler için, yerel olarak dağıtılmış bir mobil bilgi işlem sistemi içeren MeDNN adı verilen özelleştirilmiş, geliştirilmiş bir bölümlendirme ve dağıtım yaklaşımı önerilmektedir (Mao vd., 2017). Önerilen yöntem, DNN modellerini mobil cihazlar arasında bölümlendirmek için Açgözlü İki Boyutlu Bölümlendirme (GTDP) algoritmasını kullanarak bireysel kaynak kısıtlamalarına uyarlanabilir yanıtlar sağlamaktadır. Bu, çıkarım hızını 1,86 ila 2,44 faktörüyle iyileştirmektedir. Başka bir çalışma, DNN mimarilerinde optimum bölme noktasını belirlemek için Yönlendirilmiş Döngüsüz Grafik (DAG) kullanılmasını önermekte ve böylece çıkarım süresini %40,5'e kadar iyileştirmektedir (Miao vd., 2020). Ancak, her iki çalışma da açgözlü arama stratejileri nedeniyle hesaplama dezavantajları göstermektedir. Eşik tabanlı bir iş yükü bölümlendirme algoritması olan CoEdge, uçtaki mevcut hesaplama ve iletişim kaynaklarından yararlanarak DNN çıkarımı için dinamik bir iş yükü bölme stratejisi önermektedir (Zeng vd., 2021). Önerilen yöntem, %25,5 ile %66,9 arasında enerji tasarrufu sağlamaktadır. Ancak, bellek ve ağ kısıtlamaları nedeniyle gerçek zamanlı uygulamalarda sistem çökmelerinden muzdarip olmaktadır. Bu sorunu ele almak için, Yinelemeli Alternatif Optimizasyon (IAO) yöntemi, DNN katmanları için optimum bölme noktasını belirlemek üzere hesaplama ve iletişim maliyetlerini değerlendirmektedir (Tang vd., 2021). Ek olarak, başka bir çalışma, karmaşık ağ mimarilerini daha küçük alt modellere bölerek basitleştirmek için kanal bazında budama sunmaktadır (Yu vd., 2024). Ancak, tüm bu yaklaşımlar bellek tüketimi ve enerji verimliliğiyle ilgili zorluklarla karşılaşmaya devam etmektedir. Bu nedenle, mevcut literatürde ilk kez, MH algoritmaları, DNN'lerde bölme katmanını belirlemek için yeni bir yaklaşım olarak önerilmekte ve potansiyel olarak daha verimli bir çözüm sunmaktadır. Ek olarak bu çalışmalar veri

paketi kaybını ele almamaktadır. Bu veri kayıpları genellikle çeşitli veri yeniden yapılandırma teknikleri kullanılarak azaltılmaktadır.

1.3. İŞBİRLİKÇİ ZEKA VE VERİ YENİDEN YAPILANDIRMA

CI alanında, görüntü paketleri E2E-E2C arasında iletilirken kayba maruz kalmaktadır. Bu sorun, özellik tensörleri içindeki eksik veri paketlerini kurtarmak için tensör tamamlama yöntemlerinin incelenmesinde önemli ilgi görmektedir. (Liu vd., 2013)'de yazarlar, basit düşük rütbeli tensör tamamlama (SiLRTC) ve yüksek doğruluklu düşük rütbeli tensör tamamlama (HaLRTC) dahil olmak üzere temel yöntemleri vurgulamaktadır. SiLRTC, bağımlılıkları ele almak için bir blok koordinat iniş yöntemi ve bir gevşetme yaklaşımı kullanarak optimum sonuçlar için potansiyelini göstermektedir. Öte yandan HaLRTC, düşük rütbeli tensör zorluklarına odaklanan alternatif bir yön yöntemi kullanmaktadır. Ancak, her iki yöntemde de yeniden yapılandırma süreleri uzun olmakta ve iyileştirme başarısı verimli olmamaktadır. Daha sonraki bir çalışmada (Bragilevsky & Bajic, 2020), yazarlar Uyarlamalı Doğrusal Tensör Tamamlama (ALTeC) adı verilen yeni bir yaklaşım tanıtılmaktadır. ALTeC, hız açısından SiLRTC ve HaLRTC gibi geleneksel yöntemlerden daha iyi performans göstermesine ve karşılaştırılabilir doğruluğu korumasına rağmen bir sınırlamayla birlikte gelmektedir. Özellikle, ALTeC, belirli bir ara özellik tensörü için uyarlanmış belirli bir DNN modelinin ön eğitimini gerektirmektedir. Bu sorunu iyileştirmeye odaklanan başka bir çalışmada (Dhondea vd., 2021), derin özellik tensörlerindeki eksik veri paketlerini kurtarmak için içerik uyarlamalı doğrusal tensör tamamlama (CALTeC) yöntemi önerilmektedir. Önerilen yöntem, verimli veri paketi geri yüklemesi için özellik tensörünün benzer kanallarındaki eksik ve kullanılabilir özellik paketleri arasındaki afin ilişkiden yararlanmaktadır. Daha da önemlisi, CALTeC hızlıdır, uyarlanabilirdir, ön eğitim gerektirmeden işbirlikçi zekada mevcut veri paketi kurtarma yöntemlerinden daha iyi performans göstermektedir. Ancak, uzak taslak görüntülerinde kanıtlanmamıştır. Başka bir çalışmada (Bajić, 2021), yazarlar, özellik tensörlerindeki eksik veri paketlerini kurtarmak için kısmi diferansiyel denklem tabanlı görüntü boyama yöntemleri kavramını, özellikle Navier-Stokes yaklaşımını (Bertalmio vd., 2001) kullanmaktadır. Böylece, Navier-Stokes boyama yöntemi, eksik bölgeleri doldurmak

için uzamsal olarak bitişik verilerden gelen yüzey akışına dayanmaktadır. Ancak bu yaklaşım görüntülerde bulanıklığa neden olmaktadır. Bu sorunun DNN tahminleri üzerindeki etkisi göz önüne alındığında iletimi optimize etmek için alternatif bir yaklaşım önermektedir (Y. Wang vd., 2021). Önerilen yöntem, buluta aktarımları sırasında temel piksellere öncelik vererek video karelerini yeniden oluşturmaktadır. Ancak, sürekli değişen trafik koşullarına hızlı adaptasyon ihtiyacı nedeniyle bu yaklaşımın önemli bir dezavantajı ise video karelerinin büyük bir kısmının kritik özellikler içermesi ve bu tür dinamik ortamlarda genel verimliliğini sınırlamaktadır. Genel olarak, bu kategorideki çalışmalar uzun mesafeli görüntülerde sınırlı başarıya sahip olmakta ve bu durum DNN tahminlerinin performansını azaltmaktadır.

1.4. İŞBİRLİKÇİ ZEKA VE OTOMATİK KODLAYICI

AE, iletim sırasında kaybolan veri paketlerini yeniden yapılandırmada kritik bir rol oynayarak veri bütünlüğünü sağlamakta ve dinamik ağ koşullarında tahmin doğruluğunu korumaktadır (Jankowski vd., 2020). AE mimarisi, veri yeniden yapılandırma yöntemlerinin neden olduğu sorunları iyileştirmek için uygulanacaktır. AE'ler genellikle giriş yeniden yapılandırma görevleri için kullanılmakta ve bir kodlayıcı ve bir kod çözücü olmak üzere iki ana bileşenden oluşmaktadır. Kodlayıcı, bir görüntü gibi bir girdiden temel özellikleri çıkarmaktan ve bunları yoğun, gizli bir gösterime sıkıştırmaktan sorumlu olmaktadır (Yao vd., 2020). Bu sıkıştırılmış form, girişi anlamak ve yeniden yapılandırmak için gereken temel bilgileri yakalamaktadır (Pathak vd., 2016; Sbai vd., 2021). Kod çözücü daha sonra bu gösterimi kullanarak orijinal verileri yeniden yapılandırmakta veya doldurmaktadır. Bu strateji, eksik pikselleri veya eksik veya bozulmuş verilerdeki ayrıntıları kurtarmak için özellikle değerli olmaktadır (Xiang vd., 2023). Genellikle önemli kaynak kısıtlamalarıyla karşı karşıya kalan IoT sistemlerinde, AE'leri kullanan CI tekniklerinin dağıtımı, optimum olmayan kaynak kullanımına yol açabilmektedir. Sonuç olarak, AE'ler cihazlar arasında daha verimli iletim için verileri sıkıştırmak için kullanılmaktadır. Bu bağlamda, (Jankowski vd., 2020), tam girdiler yerine özellik haritalarını ileterek bant genişliği tüketimini azaltan AE tabanlı Derin Ortak Kaynak-Kanal Kodlaması (DeepJSCC) yöntemini tanıtılmaktadır. Özellik haritası (ing: feature map) boyutlarını azaltmak, doğası gereği bir veri sıkıştırma yöntemi gibi davranarak bant genişliği

tüketimini etkili bir şekilde azaltmaktadır. Başka bir çalışma (Yao vd., 2020), AE'lerin E2C boşaltmadaki rolünü araştırmakta ve AE tarafından kolaylaştırılan veri sıkıştırma, mevcut bant genişliğine göre veri işleme için optimum bölme noktalarını belirlemek için kullanılmaktadır. Bu yöntem, hesaplama görevlerinin ve veri iletiminin dağıtımını optimize ederek, IoT sistemleri içindeki sınırlı kaynakların verimli kullanımına daha fazla katkıda bulunmaktadır. Bu çalışmalarda kaydedilen ilerlemeye rağmen, genellikle iletilen toplam veri hacmi ve iletim sırasında veri kaybı potansiyeli ile ilgili sorunları tam olarak ele almamaktadır. Bunu ele almak için, (Sbai vd., 2021)'te tanımlanan araştırma, düşük güçlü geniş alan ağı (LoRa) gibi düşük bant genişliği teknolojileriyle donatılmış sistemlerde AE'lerin kullanımını araştırmaktadır. Bu bağlamda, AE'ler bölünen nokta algılama ve veri iletimini yöneterek düşük ağ koşulları altında karmaşık DNN'lerin işleyişini desteklemektedir. Ancak, bu yöntem bölünen noktaları dinamik olarak hesaplamamakta veya kablosuz iletim sırasında görüntü paketlerindeki kayıpları azaltmaya odaklanmamaktadır.

Tablo 1. Önerilen DSPCI-MTL ilgili çalışmalara genel bakış.

Reference	Görüntü Yeniden Yapılandırma	DNN Bölünme Noktası	İşbirlikçi Zeka	Çok Görevli Öğrenme	DNN Tahmin Çıktısı
(Alvar & Bajic, 2019)	+	-	+	+	+
(Karjee, Naik S, vd., 2022)	-	+	Nicemleme Olmadan	+	-
(Karjee, Anand, vd., 2022)	-	+	Nicemleme Olmadan	-	-
(Bakhtiarnia vd., 2023)	-	+	Nicemleme Olmadan	-	-
(Hu vd., 2024)	-	-	+	-	+
(Alvar vd., 2022)	+	-	+	+	+
(Capogrosso vd., 2023)	-	+	+	-	+
(Liu vd., 2013)	+	-	-	-	+
(Bragilevsky & Bajic, 2020)	+	-	+	-	+
(Dhondea vd., 2021)	+	-	+	-	+
(Bajić, 2021)	+	-	+	-	+
(Y. Wang	+	-	+	-	+

vd., 2021)					
(Jankowski vd., 2020)	-	-	+	-	+
(Yao vd., 2020)	+	-	+	-	+
(Sbai vd., 2021)	-	+	+	-	+

Tablo 2. Önerilen DSPCI-MH ilgili çalışmalara genel bakış.

Referans	Çok Görevli	Bölünme Noktası Belirleme	Enerji	Hafıza	Çıkarım Süresi	Batch Boyutu	Bant Genişliği
(Alvar & Bajic, 2019)	+	-	-	-	+	-	-
(Alvar vd., 2022)	+	-	-	-	+	-	-
(Kang vd., 2017)	-	-	+	-	+	-	+
(Eshratifar vd., 2021)	-	+	+	-	+	+	-
(Li vd., 2020)	-	+	-	-	+	-	+
(H. Wang vd., 2019)	-	+	-	-	+	-	+
(Gao vd., 2019)	-	+	-	-	-	-	+
(Capogrosso vd., 2023)	-	+	-	-	+	-	+
(Bakhtiarnia vd., 2023)	-	+	-	-	+	+	+
(Karjee, Naik S, vd., 2022)	-	+	+	-	+	-	+
(Karjee, Anand, vd., 2022)	-	+	-	-	+	-	-
(Mao vd., 2017)	-	+	-	+	-	-	-
(Miao vd., 2020)	-	+	-	-	+	-	-
(Zeng vd., 2021)	-	+	+	-	+	-	+
(Tang vd., 2021)	-	+	-	-	+	-	+
(Yu vd., 2024)	-	+	+	+	+	-	-

İKİNCİ BÖLÜM

2. Önerilen MTL-Tabanlı Metot

Bu bölümde, IoT ve DNN uygulamalarındaki iki temel zorluk olan rastgele belirlenen bölünme noktaları ve özellik haritalarının bulut cihaza iletilmesi sırasında oluşan veri kaybının iyileştirilmesi için önerilen metodoloji ayrıntılı olarak sunulmakta ve gerçekleştirilen deneyler doğrultusunda elde edilen bulgular değerlendirilmektedir. Öncelikle, geliştirilen yöntemin temel bileşenleri, teorik altyapısı ve uygulama aşamaları ele alınmakta, ardından kapsamlı deneysel analizler ışığında yöntemin etkinliği ve performansı tartışılmaktadır. Elde edilen sonuçlar, yöntemin IoT sistemlerinde tahmin doğruluğunu artırma ve işlem sürelerini optimize etme konusundaki katkılarını ortaya koymaktadır.

2.1. ÖNERİLEN DSPCI-MTL METODOLOJİSİ

Bu bölüm, IoT sistemlerinde E2E-E2C cihazları için görüntü işleme uygulamalarında tahmin doğruluğunu artırmak ve çıkarımı hızlandırmak için geliştirilen metodolojiyi tanıtmaktadır. Bu araştırmanın merkezinde, tek başına IoT cihazlarının belirli özelliklerinden ziyade etkili bulut analizine odaklanan bu cihazlar ile bulut arasındaki etkileşim yer almaktadır. Bu nedenle, cihaz özelliklerinden bağımsız olarak genel bir yöntem önerilmektedir. İşbirlikçi Zeka ile Çoklu Görev Öğrenme Uygulamasında Dinamik Bölünme Nokta Hesaplaması (DSPCI-MTL) olarak adlandırılan yöntem, verimli ağ bölümlendirmesi elde etmek için MTL'i CI ile entegre etmektedir.

2.1.1. Önerilen DSPCI-MTL Mimarisi

Önerilen DSPCI-MTL metodolojisinin mimarisi Şekil 1'de gösterilmektedir. Önerilen yöntem, MTL yaklaşımını kullanarak bir DNN eğitimi ile başlamaktadır. Eğitimin ardından, mevcut ağ bant genişliğine ve buluta iletilecek veri hacmine göre dinamik bir bölünme noktası belirlenmektedir. Bu bölünme noktası, E2E-E2C cihazları tarafından üretilen çıktılar için özel olarak optimize edilmiş DNN

katmanlarına nicemleme uygulandıktan sonra tanımlanmaktadır. Bu işlem, verimli veri işlemeyi sağlamak için ağın en uygun şekilde bölümlenmesini sağlamaktadır. Bölme noktasını tanımladıktan sonra DNN dinamik olarak bölümlenmektedir. İlk segment, ham veri girişlerini işleyerek ve nicemlenen özellik haritaları üreterek uç cihazda çalışmaktadır. Bu özellik haritaları daha sonra paketlenerek buluta iletilmektedir. Nicemleme (ing: Quantization), veri temsilindeki değişiklikler nedeniyle genellikle DNN performansını düşürse de, bu sorun, verileri orijinal biçimine geri yükleyen buluttaki tersine nicemleme yoluyla iyileştirmektedir. Veri iletimi sırasında kaçınılmaz paket kayıplarını gidermek için, özellik haritalarındaki kayıp verileri yeniden oluşturmak için bir AE kullanılmaktadır. Bu yeniden yapılanma, DNN'in ikinci segmentinin, veri kaybı ve iletim kısıtlamalarının zorluklarına rağmen yüksek performansı koruyarak, belirli uygulamalara uyarlanmış tahmin görevlerini etkin bir şekilde yürütebilmesini sağlamaktadır. Önerilen yaklaşım için bu bölümde DNN eğitiminin aşamaları, CI-MTL ve Dinamik Bölünme Nokta Uygulaması ve AE verilerinin yeniden yapılandırılması açıklanacaktır. Deneysel bulgular DNN mimarisinin bir açıklamasını sağlayacaktır

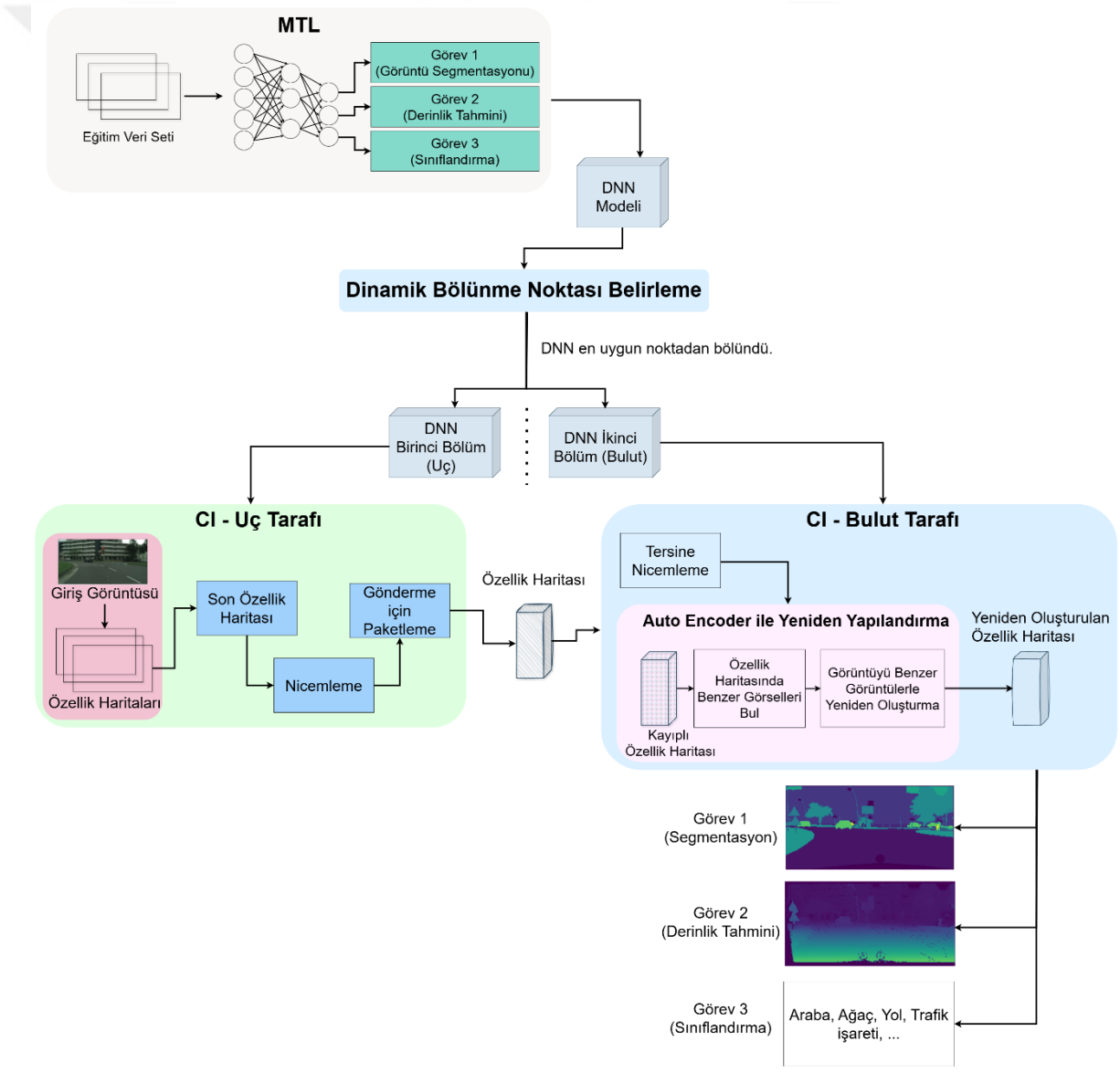
Önerilen yöntemin temel amaçları gecikmeyi azaltmak, doğruluğu artırmak ve kaynak kullanımını verimli hale getirmektir. Bu bağlamda, uç ve bulut arasındaki iş yükünü dengeleyerek işleme süresini optimize etmekte ve gecikmeleri azaltmaktadır. Ek olarak, AE tabanlı veri rekonstrüksiyonu ile tahmin doğruluğu korunarak artırılmaktadır. Bu hedefler doğrultusunda önerilen yöntemin çalışma mekanizmasını şu şekilde özetleyebiliriz.

1) DNN ve MTL Entegrasyonu: Bu yaklaşım, DNN'lerin sınıflandırma, bölümleme ve derinlik tahmini olmak üzere aynı anda birden fazla görevi öğrenmesini sağlamaktadır. MTL, görevler arasında ortak temsil öğrenimini teşvik ederken modelin genel verimliliğini artırmaktadır.

2) Uç ve Bulut Cihazları Arasındaki İşbirliği: CI çerçevesinde, modelin bir bölümü uç cihazda işlenmek üzere ayrılırken diğer bölümü bulutta çalıştırılmak üzere yapılandırılmaktadır. Bu yaklaşım, ilk işlemleri uç cihazda gerçekleştirerek veri mahremiyetini korumaktadır. Ayrıca, daha hesaplama yoğun işlemlerin bulut tarafında yürütülmesi sayesinde işlem süresi ve kaynak kullanımı optimize edilmektedir.

3) Dinamik bölünme noktası belirleme: Modelin bölünme noktası, cihazların ağ bant genişliğine, veri hacmine ve işlem kapasitesine göre dinamik olarak belirlenmektedir. Bu, ağın iş yükünü uç ile bulut arasında en verimli şekilde paylaşmasını sağlamaktadır. Bölme noktası belirleme işlemi, katmanlardaki veri yoğunluğu ve sıkıştırma oranlarına dayanmaktadır.

4) AE ile veri yeniden yapılandırma: Uç cihazdan buluta veri aktarımı sırasında oluşabilecek veri kaybını telafi etmek için bulutta bir AE kullanılmaktadır. AE, yeniden yapılandırma işlevlerini devralmakta ve kayıp görüntü verilerini yeniden yapılandırarak sistemin tahmin doğruluğunu artırmaktadır.



Şekil 1. Önerilen DSPCI-MTL Metodolojisinin İllüstrasyonu.

2.1.2. CI-MTL & Dinamik Bölünme Noktası Uygulaması

Daha önce tartışıldığı üzere CI, hesaplama verimliliğini artırmak ve kaynak kullanımını optimize etmek için DNN'yi birden fazla cihaza bölmeyi içermektedir. Bu strateji, sürü sistemleri veya E2E-E2C cihazları gibi karmaşık sistemlerin çalışmasını kolaylaştırmakta ve paylaşılan bir DNN'nin farklı cihazlarda çalışmasına olanak tanımaktadır. DNN'yi bölerek, uç cihazlar buluta görüntü verileri yerine özellik haritaları ileterek veri gizliliğini artırmakta ve bant genişliği kullanımını azaltmaktadır. Önerilen CI-MTL metodolojisi, paylaşılan bir mimaride ilgili görevleri ortak olarak eğiterek öğrenme verimliliğini ve performansını artırmak için MTL'den yararlanmaktadır. Bu yaklaşım, yalnızca görevler arasında genellemeyi iyileştirmekle kalmaz, aynı zamanda gösterimleri paylaşarak hesaplama maliyetini de optimize etmektedir. Uygulamamız üç temel göreve odaklanır:

- Derinlik tahmini (T^{de})
- Görüntü segmentasyonu (T^{s})
- Görüntü sınıflandırması (T^{cl})

Her görev, geliştirilen üç birincil sınıflandırıcı olan $'T'$ sınıfı altında $'T^{d}'$, $'T^{s}'$ ve $'T^{cl}'$ olarak kategorize edilmektedir. Görev sınıflandırıcıları, her sınıflandırıcı k'nın ilgili görevi için öğrenmeyi optimize edecek şekilde uyarlandığı veri kümesi boyunca eğitilmektedir. Sınıflandırıcı indeksi $\{T_i^k\}_{i=1}^m$, olarak gösterilmektedir. Burada $'i'$ görev indeksini ve *'m'* maksimum öğrenme görevi sayısını belirtmektedir. Bu yapı, her görevin en iyi şekilde öğrenilmesini sağlayarak veri örneklerinden etiketlere işlevsel eşlemeleri verimli bir şekilde modellemektedir (Y. Zhang & Yang, 2018). Sınıflandırıcılar arasında kullanılan her bir görev için eğitim kümesi denklem 1'de gösterilmektedir.

$$D_i = \{(x_j, y_j)\}_{j=1}^n \tag{1}$$

Burada $'n'$ her görev için mevcut eğitim örneklerinin sayısını ve $'j'$ her örnek endeksini belirtmektedir. (x_j, y_j) sırasıyla örnekler ve etiketleri temsil etmektedir. Bu görevler arasında kullanılan genelleştirilmiş öğrenme fonksiyonu denklem 2 ile hesaplanmaktadır.

$$M(x_j, y_j) = train\left\{T_i^k(x_j, y_j)\right\}_{i=1}^{m} \quad (2)$$

Burada *'m'*, ağ genelinde kapsamlı öğrenmeyi garanti eden maksimum öğrenme görevi sayısını temsil etmektedir. Veri kümesi, MTL aracılığıyla uygun sınıflara kategorize edilerek eğitildikten sonra ağ kritik bir faz-dinamik bölme noktası belirlemesinden geçmektedir. Bu süreç, sınırlı kapasiteli donanım platformlarında verimli işlemeyi kolaylaştırarak, tensörleri daha düşük bit genişliği depolama için uyarlamak amacıyla nicemleme içermektedir (Choi & Bajic, 2018). Nicemleme ve sonraki bölme noktası hesaplaması formülü denklem 3'e dayanmaktadır.

$$V_e = round\left(\frac{V_e - min(V_e)}{max(V_e) - min(V_e)} - (2^{n_{bit}} - 1)\right) \quad (3)$$

Burada, *'e'* tensörün her katmanını ifade etmektedir. $V_e \in \mathbb{R}^{R \times C \times Ch}$, her katmandaki özellik haritası verilerini içeren tensör olmakta ve *'R'*, *'C'* ve *'ch'* sırasıyla satırların, sütunların ve kanalların boyutlarını temsil etmektedir. Nicemlemenin ardından, her katmanın yapılandırması, optimum bölme noktasının belirlenmesini bildirmektedir. Bu yapılandırma, denklem 4'te bulunan fonksiyonla tanımlanmaktadır.

$$f(x_j) = f_L(f_{L-1}(\cdots f_1(x_j))) \quad (4)$$

Burada, x_j girişi belirtirken, 'f_1' ile 'f_L' arasındakiler ağın girişinden çıkışına kadar olan işlem dizisini temsil etmektedir. *'h'* olarak belirtilen katmandan gelen özellik haritası çıktısı, verilerin uçtan (ön taraf) buluta (arka taraf) geçişini belirtmektedir. Bölünme noktalarının en iyi şekilde belirlenmesi ayrıca ağın her bir parçası için sıkıştırma oranlarının ve çıkarım sürelerinin analizini içermektedir. Veri verimliliğini değerlendirmek için kritik olan sıkıştırma oranı denklem 5'e göre hesaplanmaktadır.

$$c_L = \frac{|V(h_L)|}{x} \quad (5)$$

Burada *'$V(h_L)$'*, *'L'* katmanının nicelleştirilmiş çıktısı ve *'x'*, o katmana ait girdi boyutunu ifade etmektedir. Optimum bölme noktaları için çıkarım süreleri, denklem 6 kullanılarak belirlenmektedir.

$$IT_z = IT_{1,z}^f + c_z IT_{FO} + IT_{z+1,k}^b + IT_{k+1,L}^b \quad (6)$$

Burada '$IT_{1,z}^{f}$' birinci katmandan z^{th} katmana kadar kümülatif çıkarım süresi olmakta ve '$IT_{z+1,k}^{b}$' uç cihazdan bulut cihaza kadar olan işlem süresini temsil etmektedir. '$IT_{k+1,L}^{b}$' son katmana kadar arka taraftaki DNN alt modelinin çıkarım süresi olarak tanımlanmaktadır. '$c_z IT_{FO}$' bu katmanın sıkıştırma oranını belirtmektedir. 'IT_{FO}' bölünme noktasının olmadığı tam boşaltma senaryosundaki iletim süresini temsil etmektedir. Bant genişliği ve veri hacmindeki dalgalanmalar dikkate alındığında toplam uçtan uca çıkarım süresi denklem 7 ile hesaplanmaktadır.

$$IT_z = IT_{1,z}^{f} + \frac{Dc_p}{r} + IT_{z+1,L}^{b} \tag{7}$$

Burada, 'Dc_p' sıkıştırma ve iletim için hazırlanan verileri, 'r' mevcut bant genişliğini ve '$\frac{Dc_p}{r}$' ise 'z^{th}' katmandaki sıkıştırma etkisini temsil etmektedir. '$IT_{z+1,L}^{b}$' ifadesi, z + 1 katmanından L katmanına kadar olan çıkarım süresini ölçmektedir. Diğer terimler ise daha önce tanımlanmıştır. Daha sonra denklem 8'e göre tüm olası katmanlarda bu çıkarım süresini en aza indirerek optimum bölme noktası olan 'S_{opt}' belirlenmektedir.

$$s_{Opt} = \underset{z \in \{0,L\}}{\operatorname{argmin}}\, IT_z \tag{8}$$

Bu, gerçek zamanlı ağ koşulları ve iş yükü gereksinimlerine bağlı olarak hesaplama görevlerinin uç ve bulut arasında etkili bir şekilde dağıtılmasıyla DNN'nin optimum şekilde çalışmasını sağlamaktadır.

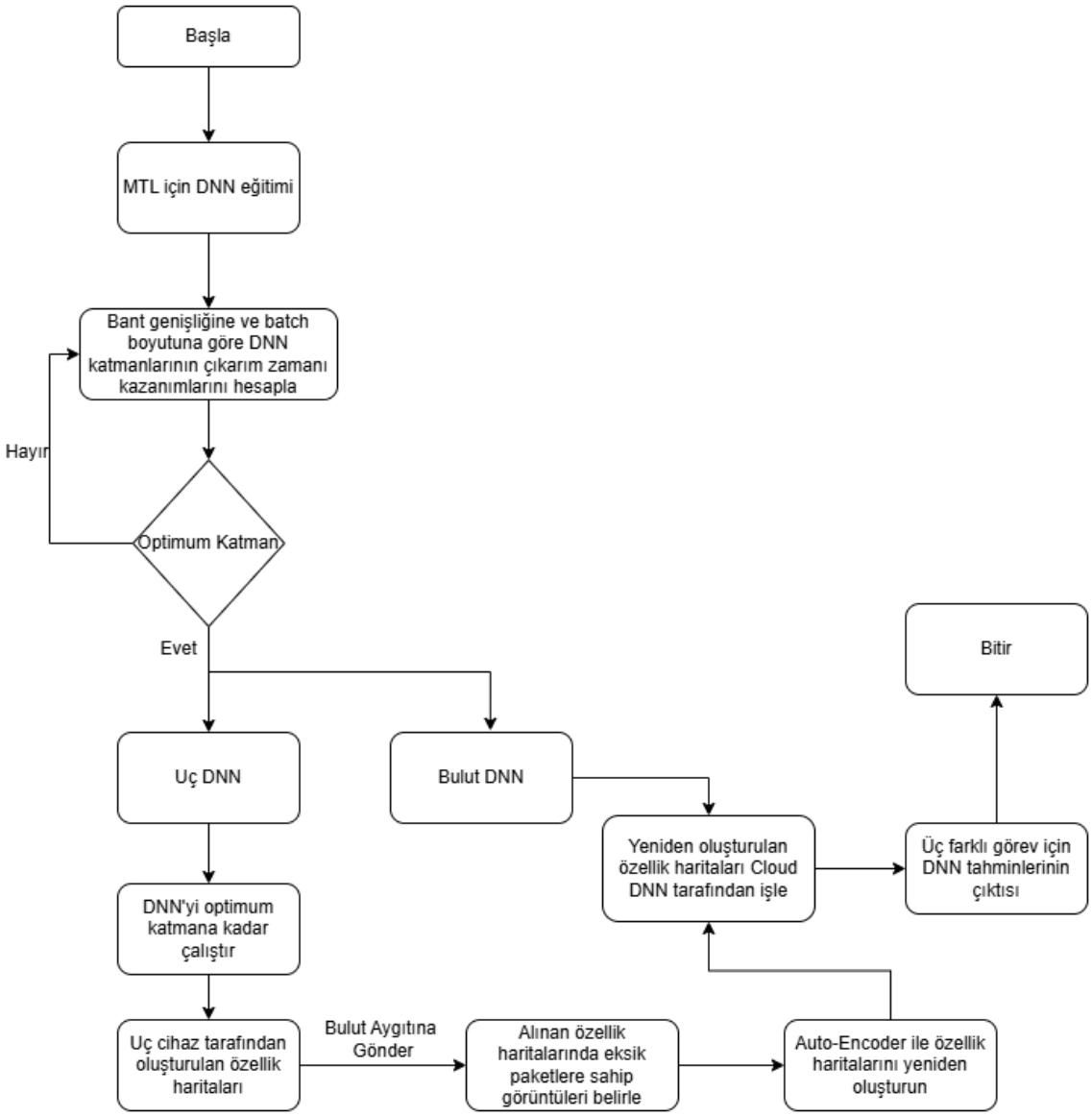
2.1.3. Auto Encoder ile Veri Yeniden Yapılandırma

CI kullanan bulut tabanlı sistemlerde, nicemleme ve iletim aşamaları sırasındaki veri kayıpları, tahmin modellerinin performansını önemli ölçüde düşürebilmektedir (Choi & Bajic, 2018; Eddahmani vd., 2023). Bu makale, bulut tarafında tahmin doğruluğunu korumak için gerekli olan bu tür kayıplardan etkilenen veri paketlerini yeniden oluşturmak için AE mimarilerinin etkinliğini araştırmaktadır. AE, öncelikle ek sıkıştırma gerektirmeden doğal bir veri darboğazı görevi görerek kodlama ve kod çözme işlemleri için kullanılmaktadır. Bu işlev, ağ kaynaklarının verimli bir şekilde kullanılmasını kolaylaştırdığı ve uç ve bulut cihazları arasındaki iletimlerde veri bütünlüğünü koruduğu CI sistemlerinde özellikle avantaj sağlamaktadır (Aljabri vd., 2024; Matsubara vd., 2023; Uyanik vd., 2023).

Geleneksel olarak, AE'ler CI çerçeveleri içinde doğrudan görüntü yeniden oluşturma için yaygın olarak uygulanmamaktadır. Ancak önerilen çalışmada, özellikle nicemleme sonrası görüntü kurtarma ve iletim hatalarını hedef alarak bu kapasitedeki potansiyellerini araştırmaktadır. Yeniden yapılandırma süreci, görüntülerin kalitesini parlaklık, kontrast ve yapısal benzerlik olmak üzere üç temel unsura göre değerlendiren Yapısal Benzerlik Endeksi Ölçümü'nden (SSIM) yararlanmaktadır. Bu metrikler, benzerliği değerlendirmek için SSIM endeksinde (Nilsson & Akenine-Möller, 2020) birleştirilen görüntülerin ortalama parlaklığı, varyansı ve kovaryansı kullanılarak hesaplanmaktadır. SSIM endeksi, AE eğitimi için orijinal, kayıp öncesi görüntülere en çok benzeyen özelliklere sahip görüntüleri seçmeye yardımcı olmaktadır. Bu hedefli eğitim, AE'lerin kritik görüntü özelliklerini etkili bir şekilde öğrenmesini ve yeniden yapılandırmasını sağlayarak, kaybolan veri paketlerinin yüksek kalitede kurtarılmasını sağlamaktadır. AE'leri bu yenilikçi şekilde uygulayarak, bu tez CI sistemlerinin sağlamlığını artırarak bulut bilişim ortamlarında güvenilir görüntü işleme sonuçları sağlamakta ve kapsamlı veri iletimi ihtiyacını azaltarak bant genişliği kullanımını optimize etmektedir. Bu yöntem yalnızca veri paketi kaybının etkisini azaltmakla kalmaz, aynı zamanda daha sürdürülebilir ve verimli ağ kaynak yönetimine de katkıda bulunmaktadır.

2.1.4. Önerilen DSPCI-MTL Yöntemin Akış Şeması ve Pseudocode

Bu bölümde, önerilen yöntemin çalışma mekanizmasını özetleyen akış şeması ve ayrıntılı sözde kod sırasıyla Şekil 2'de ve Algoritma 1'de sunulmaktadır.



Şekil 2. Önerilen DSPCI-MTL yöntemin akış şeması.

Algoritma 1. Önerilen metodun Pseudo code olarak gösterimi.

```
# Modelin nicemleme uygulanması
function Quantize_Model(M):
    Q_M = Quantize(M)  # Modelin nicemlemesini uygula
    return Q_M

# Doğal darboğazları bul
function Get_Natural_Bottlenecks(Q_M, input_size):
    Natural_Bottlenecks = []
    for L_i in Q_M.Layers[:-1]:  # Son katman hariç tüm katmanlar üzerinde döngü
        if L_i.output_shape is not None:
            Output_Size = Calculate_Output_Size(L_i.output_shape)  # Çıktı boyutunu hesapla
    return Natural_Bottlenecks

# Çıkarım süresini hesapla
function Get_Inference_Time(Model, Batch_Size, Input_Shape, Repetitions=10):
    Warmup(Model, Batch_Size, Input_Shape, Repetitions)  # Isınma iterasyonları
    Start_Time = Timer()
    for i in range(0, Repetitions):
        Model(Predict(Input_Shape))
    End_Time = Timer()
    Inference_Time = (End_Time - Start_Time) / Repetitions
    return Inference_Time * 100  # Milisaniye cinsinden döndür

# Belirli bir katmandan itibaren kuyruk modelini çıkar
function Get_Tail_Model(M, Start_Layer_Index):
    Tail_Model = Subset_Model(M, Start_Layer_Index, End_Layer=M.End)
    return Tail_Model

# Her bölme noktası için çıkarım süreleri ile bir batch tablosu oluştur
function Create_Batch_Table(M, Config):
    Q_M = Quantize_Model(M)
    Natural_Bottlenecks = Get_Natural_Bottlenecks(Q_M, Config.input_size)
    Batch_Table = {}
    for Batch_Size in Config.batch_sizes:
        Batch_Table[Batch_Size] = {}

# Uç cihazda tüm model için çıkarım süresini hesapla
        with Device(Config.processors.weak):
            Batch_Table[Batch_Size]["Whole_Device"] = Get_Inference_Time(Q_M, Batch_Size, Config.input_shape)
        # Bölmeler için çıkarım süresini hesapla
        for N in Natural_Bottlenecks:
            Head_Model = Extract_Head_Model(Q_M, N["Layer_Name"])
            with Device (Config.processors.weak):
                Head_Time = Get_Inference_Time(Head_Model, Batch_Size, Config.input_shape)
            Next_Layer_Index = Get_Layer_Index(Q_M, N["Layer_Name"]) + 1
            Tail_Model = Get_Tail_Model(Q_M, Next_Layer_Index)
            with Device(Config.processors.strong):
                Tail_Time = Get_Inference_Time(Tail_Model, Batch_Size, Get_Layer_Input_Shape(Tail_Model))
            Data_Load = Calculate_Load(N["Bandwidth"], Batch_Size, Config)
            Total_Time = Head_Time + Tail_Time + (Data_Load / Config.bandwidth)
            Batch_Table[Batch_Size][N["Layer_Name"]] = {
                "Total_Time": Total_Time
            }
    Save_Batch_Table(Batch_Table, Config)
    return Batch_Table

# Çıkarım sonuçlarını analiz et ve grafikle
function Create_Inference_Plots(Batch_Table, Config):
    Split_Points = Get_Split_Points(Batch_Table)
    Bandwidths = Config.bandwidth_range
    for Batch_Size in Batch_Table.keys():
        Results = Compute_Times_For_Splits(Batch_Table[Batch_Size], Bandwidths)
        Plot_Inference_Times(Results, Bandwidths, Batch_Size)
    Analyze_Optimal_Splits(Batch_Table)
    Visualize_Optimal_Splits(Batch_Table, Config)
```

2.2. ÖNERİLEN DSPCI-MTL YÖNTEMİ DENEYSEL SONUÇLARI

Bu bölümde, önerilen DSPCI-MTL metodolojisinin daha çeşitli veri kümeleri üzerinde uygulanmasına odaklanılarak iki farklı veri kümesiyle elde edilen sonuçlar sunulmaktadır. Bu bağlamda, ilk veri kümesi olan Cityscapes (Cordts vd., 2016), derinlik tahmini ve anlamsal segmentasyon görevleri için piksel düzeyinde etiketlemeye sahip toplam 2975 eğitim, 500 doğrulama ve 1525 test görüntüsünden oluşmaktadır. Diğer veri kümesi ise ImageNet (Deng vd., 2009), sınıflandırma görevi için kullanılmakta ve 1000 nesne sınıfını içeren 1.281.167 eğitim, 50.000 doğrulama ve 100.000 test görüntüsünden oluşmaktadır. Cityscapes ve ImageNet veri kümelerinin temel özellikleri Tablo 3'te sunulmaktadır. Derinlik tahmini performansını değerlendirmek için Kök Ortalama Kare Hatası (RMSE) metriği kullanılmıştır. RMSE, tahmin edilen ve gerçek derinlik değerleri arasındaki doğruluğu niceliksel olarak ifade etmektedir. Daha düşük RMSE değeri, tahminlerin gerçek derinlik değerlerine daha yakın olduğunu göstermektedir (Blanc-Durand vd., 2020; Z. Wang vd., 2020). Anlamsal segmentasyon performansını değerlendirmek için Kesişim Birleşim Oranı (IoU) metriği kullanılmıştır. Bu metrik, tahmin edilen segmentasyon ile gerçek segmentasyon arasındaki örtüşmeyi ölçerek, piksel sınıflandırmasının doğruluğunu değerlendirmektedir. Tüm sınıflar için IoU değerlerinin ortalaması alınarak Ortalama IoU (Mean IoU) hesaplanmakta ve böylece hem modelin genel performansını hem de her bir sınıf için etkinliğini ölçmek amacıyla kullanılmaktadır (Pham, 2021). Sınıflandırma görevinde ise modelin doğru tahmin performansını değerlendirmek için doğruluk (accuracy) metriği kullanılmaktadır (Mrudula vd., 2024). Bu metriklerle, bir sınıfın doğru tahmin edilip edilmediği nesnel bir değerlendirme kriteri olarak belirlenmiştir.

Tablo 3. Veri kümelerinin açıklaması.

Veri Seti	Eğitim Seti	Test Seti	Doğrulama Seti	Obje Sınıf Boyutu
Cityscapes	2975	1525	500	30
ImageNet	1.281.167	100.000	50.000	1000

2.2.1. DSPCI-MTL için Kurulum ve Yapılandırma

Önerilen yöntem için deneyler, tablo 4'te ayrıntılı olarak sunulan belirli bir yazılım ve donanım yapılandırması kullanılarak gerçekleştirilmiştir. Bu yapılandırmalar, kullanılan Python ve TensorFlow sürümleriyle birlikte hem uç hem de bulut bilişim bileşenlerine ait donanım özelliklerini içermektedir. Uç cihaz olarak, düşük kapasiteli bir CPU'ya sahip bir dizüstü bilgisayar kullanılmıştır. Buna karşılık, bulut bileşeni, yüksek performanslı GPU'lar içeren uzak bir sunucu tarafından yönetilerek uç cihazın CPU'suna kıyasla önemli ölçüde daha yüksek hesaplama gücü sunmaktadır. Ayrıca, önerilen yöntemin analiz ve değerlendirilmesi için kullanılan parametreler ile karşılaştırma amacıyla belirlenen ölçütler tablo 5'te tanımlanmıştır.

Tablo 4. Kullanılan yazılım versiyonları ve donanımların özellikleri.

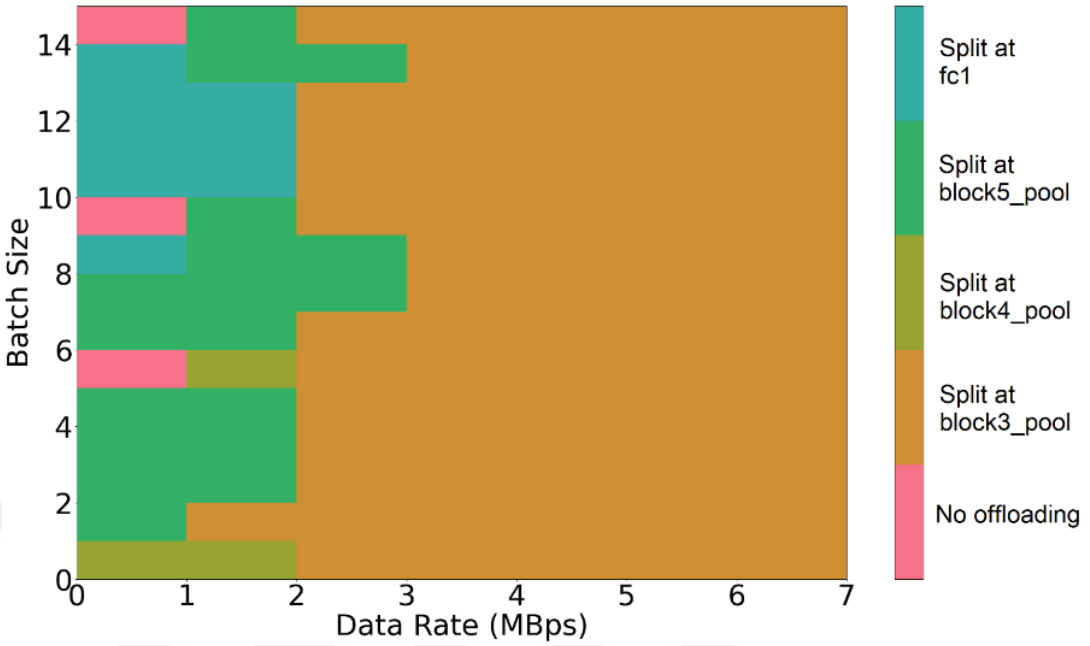
Komponent	**Uç Cihaz (Laptop)**	**Bulut Sunucu (Remote)**
CPU	Intel Core i5-7300HQ @ 2.50 GHz	Intel Xeon Gold 6226R @ 2.30 GHz
RAM	8 GB DDR4	64 GB DDR4
Depolama	256 GB SSD	2 TB NVMe SSD
GPU	Yok	NVIDIA Tesla T4 (16 GB)
İnternet Ağı	Wi-Fi (50 Mbps, 2.4/5 GHz)	Gigabit Ethernet (1-10 Gbps)

Tablo 5. Analiz ve değerlendirmede kullanılan parametreler.

Parametre	Açıklama
SSIM	Bu ölçüm, görüntülerin ortalama parlaklığı, varyansı ve kovaryansı dikkate alınarak hesaplanmaktadır. Bu bileşenler daha sonra tahmin edilen ve gerçek görüntüler arasındaki benzerliği değerlendirmek için SSIM ile birleştirilmektedir. SSIM indeksi, doku, kontrast ve parlaklık yapısal bilgilerine odaklanarak görüntüler arasındaki algısal farklılıkları yakalayarak görüntü segmentasyonu ve restorasyonu görevlerinde görüntü kalitesini ve yeniden yapılandırmanın doğruluğunu değerlendirmek için kullanılmaktadır.
PSNR	Tepe Sinyal-Gürültü Oranı (PSNR), orijinal ve bozulmuş sürümler arasındaki farkı ölçerek bir görüntünün veya videonun kalitesini değerlendirmek için kullanılmaktadır. Daha yüksek bir PSNR değeri, minimum bozulma veya bilgi kaybını yansıtarak orijinal ile bozuk görüntü arasındaki farkın daha küçük olduğunu göstermektedir. Bu ölçüm, görsel doğruluğu korumada algoritmaların etkinliğini değerlendirmek için basit bir yöntem sunduğundan görüntü sıkıştırma, yeniden oluşturma ve restorasyon görevlerinde yaygın olarak kullanılmaktadır.
MeanIoU	Anlamsal segmentasyonun performansını değerlendirmek için Birleşim Üzerindeki Kesişme (IoU) metriği kullanılmaktadır. Bu ölçüm, tahmin edilen ve temel gerçek segmentasyonları arasındaki örtüşmeyi ölçerek piksel sınıflandırmasındaki doğruluğun bir değerlendirmesini sağlamaktadır. Tüm sınıflar genelinde ortalama IoU'yu hesaplayan Ortalama IoU, segmentasyon modelinin genel performansını ölçmek için kullanılmaktadır. Bu yaklaşım, hem modelin genel etkililiğini hem de bireysel sınıflar arasındaki performansının ayrıntılı bir değerlendirmesine olanak tanıyarak modelin güçlü yönlerine ve belirli kategorilerde geliştirilebilecek alanlara dair içgörüler sunmaktadır.
RMSE	Derinlik tahminini değerlendirmek için tahmin edilen ve gerçek derinlik değerleri arasındaki doğruluğu ölçmek amacıyla Ortalama Karekök Hata (RMSE) metriği kullanılmaktadır. RMSE, tahmin edilen ve gerçek derinlik değerleri arasındaki ortalama kare farkların karekökünü hesaplayarak tahmin hassasiyetinin sayısal bir ölçüsünü sağlamaktadır. Daha düşük bir RMSE değeri, tahmin edilen derinlik değerlerinin gerçek değerlere daha yakın olduğunu göstererek derinlik tahminindeki daha yüksek doğruluğu yansıtmaktadır. Bu ölçüm, büyük hatalara karşı duyarlılığı nedeniyle yaygın olarak kullanılmakta ve bu durum derinlik tahmin modellerinin performansını değerlendirmede etkili bir araç haline getirmektedir.
Etiket Doğruluğu	Etiket doğruluğu, sınıflandırma görevlerindeki toplam etiket sayısına göre doğru tahmin edilen etiketlerin oranını ölçmektedir. Daha yüksek etiket doğruluğu, üstün model performansını yansıtarak modelin sınıflar arasında etkili bir şekilde ayrım yaptığını göstermektedir.
MAE	Ortalama Mutlak Hata (MAE), tahmin edilen değerler ile gerçek değerler arasındaki ortalama mutlak farkı ölçerek modelin tahminlerinin doğruluğunu değerlendirmek için kullanılmaktadır. Özellikle regresyon problemlerinde ve tahmin doğruluğunun değerlendirilmesinde yaygın olarak kullanılmaktadır.
DSSIM	Yapısal Farklılık İndeksi (DSSIM), iki görüntü arasındaki yapısal benzerliği değerlendirmektedir. SSIM değerinden farklı olarak görüntüler arasındaki yapısal farklılıkları ölçmektedir. Daha yüksek DSSIM değerleri, daha büyük yapısal farklılıkları ve dolayısıyla daha düşük benzerliği göstermektedir. Bu durumda, görüntü işleme uygulamalarında görüntü kalitesini değerlendirmek ve model performansını analiz etmek için kullanılmaktadır.
Çıkarım Süresi	E2E-E2C cihazlarının çıkarım süresi, ağ gecikmesi, veri aktarım süreleri ve sistem yükü gibi faktörler hesaba katılarak hem uç hem de bulut tarafındaki birleşik işlem süreleriyle belirlenmektedir.

2.2.2. Dinamik Bölünme Noktası Belirleme

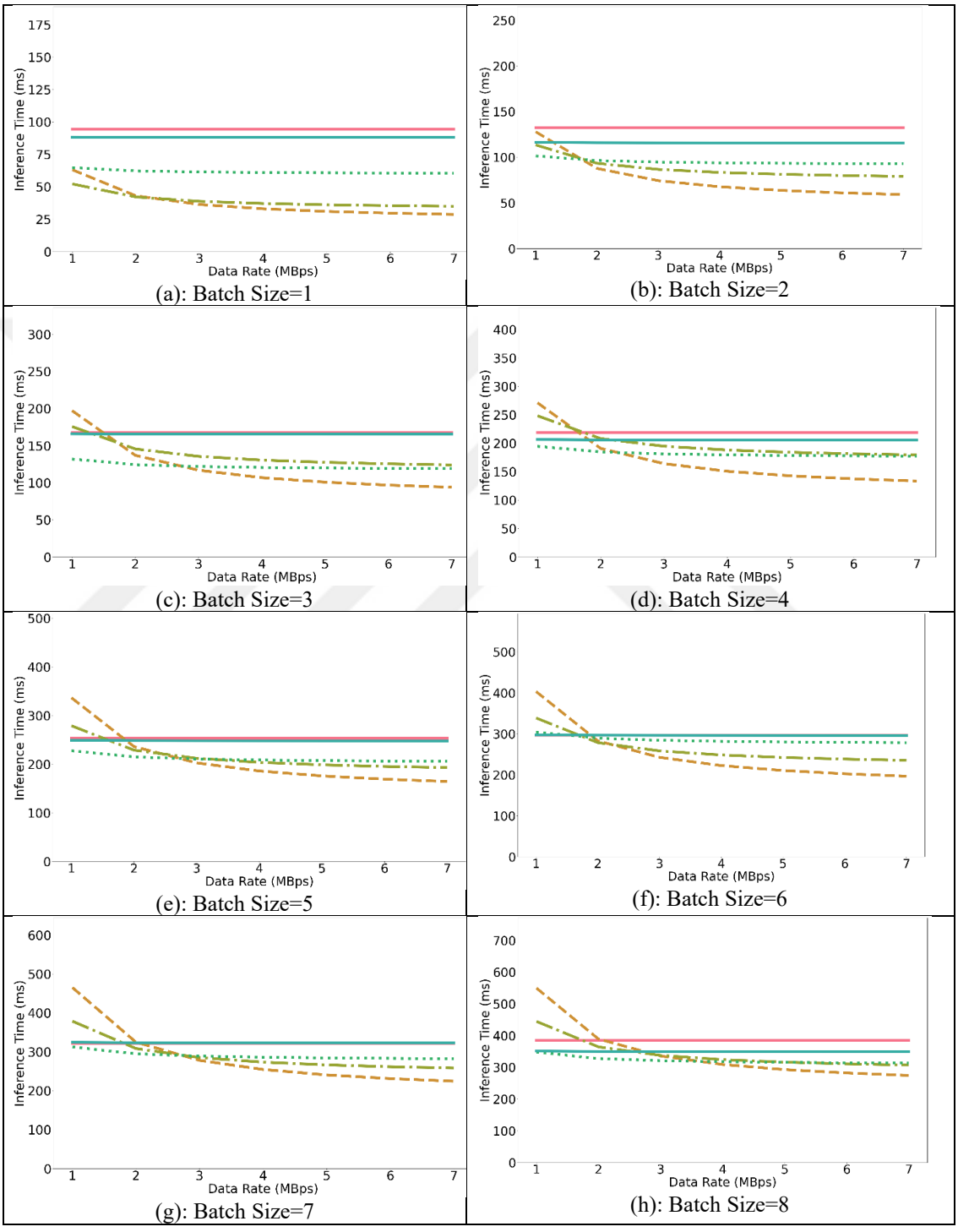
Deneylerde kullanılan DNN, VGG-16 (Simonyan & Zisserman, 2014) omurgasını temel almaktadır. Model, bir görüntüdeki her pikselin derinliğinin tahmin edildiği derinlik tahmini, piksellere belirli etiketlerin atandığı segmentasyon ve nesnenin sınıfının tespit edildiği sınıflandırma olmak üzere üç görevi yerine getirmek üzere eğitilmiştir. Görevlerin gereksinimlerine göre kullanılan kayıp fonksiyonları, derinlik tahmini için Ortalama Kare Hatası (MSE), segmentasyon için kategorik çapraz entropi ve sınıflandırma için softmax olarak belirlenmiştir (Terven vd., 2023). Önerilen metodolojinin kritik bir yönü, DNN mimarisi içinde dinamik bölme noktası belirleme sürecinin simülasyonu olarak tasarlanmıştır. Bu süreç, verilerin başlangıçta uç cihazda işlenmesini, ardından nicemleme işlemi uygulanarak nihai hesaplamanın yapılacağı buluta iletilmesini içermektedir. Bölme noktası kararı, uç cihazın buluta kıyasla daha sınırlı işlem kapasitesine sahip olduğu göz önünde bulundurularak verilmekte ve ağın genel verimliliğini optimize edecek şekilde belirlenmektedir. Düşük internet kapasitesi senaryolarını simüle etmek amacıyla bant genişliği 7 MBps ile sınırlandırılmış ve bellek taşmasını önlemek için maksimum batch boyutu 15 olarak belirlenmiştir. Bu koşullar altında, değişken veri hacimleri (1-15 batch boyutları) ve bant genişliği senaryoları (1-7 MBps) üzerinden işlem verimliliği değerlendirilerek DNN'de görev bölme için en uygun katman tespit edilmektedir. Simülasyon sonucunda, seçilen katmanın yığın boyutuna ve bant genişliğine bağlı olarak dinamik şekilde değiştiği gözlemlenmektedir. Bu doğrultuda, hangi katmanın seçildiğini gösteren ısı haritası şekil 3'te sunulmaktadır. Elde edilen sonuçlara göre, bölme katmanı her koşul altında dinamik olarak değişmektedir. Deney sırasında, batch boyutunun 15 ve bant genişliğinin maksimum olduğu durumda, en uygun bölme katmanı 'block3_pool' katmanı olarak belirlenmiştir. DNN, bu dinamik olarak seçilen katman üzerinden bölünebilmektedir. 7 MBps bant genişliği kısıtı altında dinamik olarak seçilen 'block3_pool' katmanının çıkarım süresinin değerlendirilmesi için işlenen veri hacminin dikkate alınması gerekmektedir.



Şekil 3. Veri Hacmine Bağlı Olarak Batch Boyutu ve Bant Genişliğine Göre En İyi Bölünme Noktaları.

Çıkarım sürelerini ayrıntılı olarak gösterebilmek için Şekil 4'te karşılaştırmalı grafikler sunulmaktadır. Bu grafikler, 1 ile 15 arasında değişen farklı batch boyutları altında çıkarım sürelerinin durumunu ortaya koymaktadır. Grafikte bulunan renkler farklı senaryoları temsil etmektedir. Bu durumda, pembe, işlem yükünün devredilmediği durumu belirtirken turuncu, işlemenin üçüncü havuzlama katmanına kadar gerçekleştirilmesini ve yeşil, açık yeşil ve mavi ise grafiğin lejantında belirtilen ilgili senaryoları ifade etmektedir. Bu doğrultuda, grafikteki farklı renkler, her katmanın işlem verimliliğinin değişen ağ koşulları altında nasıl farklılık gösterdiğini yansıtmaktadır. Batch boyutları üçü aştığında ve bant genişliği kısıtlı olduğunda, 'block3_pool' katmanı en iyi çıkarım sürelerini sağlamaktadır. Buna karşılık, daha yüksek bant genişliklerinde bu katman en hızlı işlem sürelerine ulaşmaktadır. Ancak, 'block4_pool' ve 'block5_pool' katmanlarının daha düşük bant genişliklerinde daha hızlı çıkarım süreleri sunduğu gözlemlenmektedir. Özellikle, ağı 'block3_pool' katmanında bölmek, tüm işlemenin uç cihazda gerçekleştirildiği konfigürasyonlara kıyasla toplam işlem süresini %38 oranında azaltmaktadır. Bu önemli iyileştirme, CI çerçevesi içinde DNN katmanlarının dinamik olarak seçilmesinin değerini vurgulamakta ve hesaplama görevlerini mevcut ağ koşulları ile

veri hacimlerine göre etkili bir şekilde dağıtarak hem uç cihazların hem de bulut bilişimin kaynaklarını optimize etmektedir.

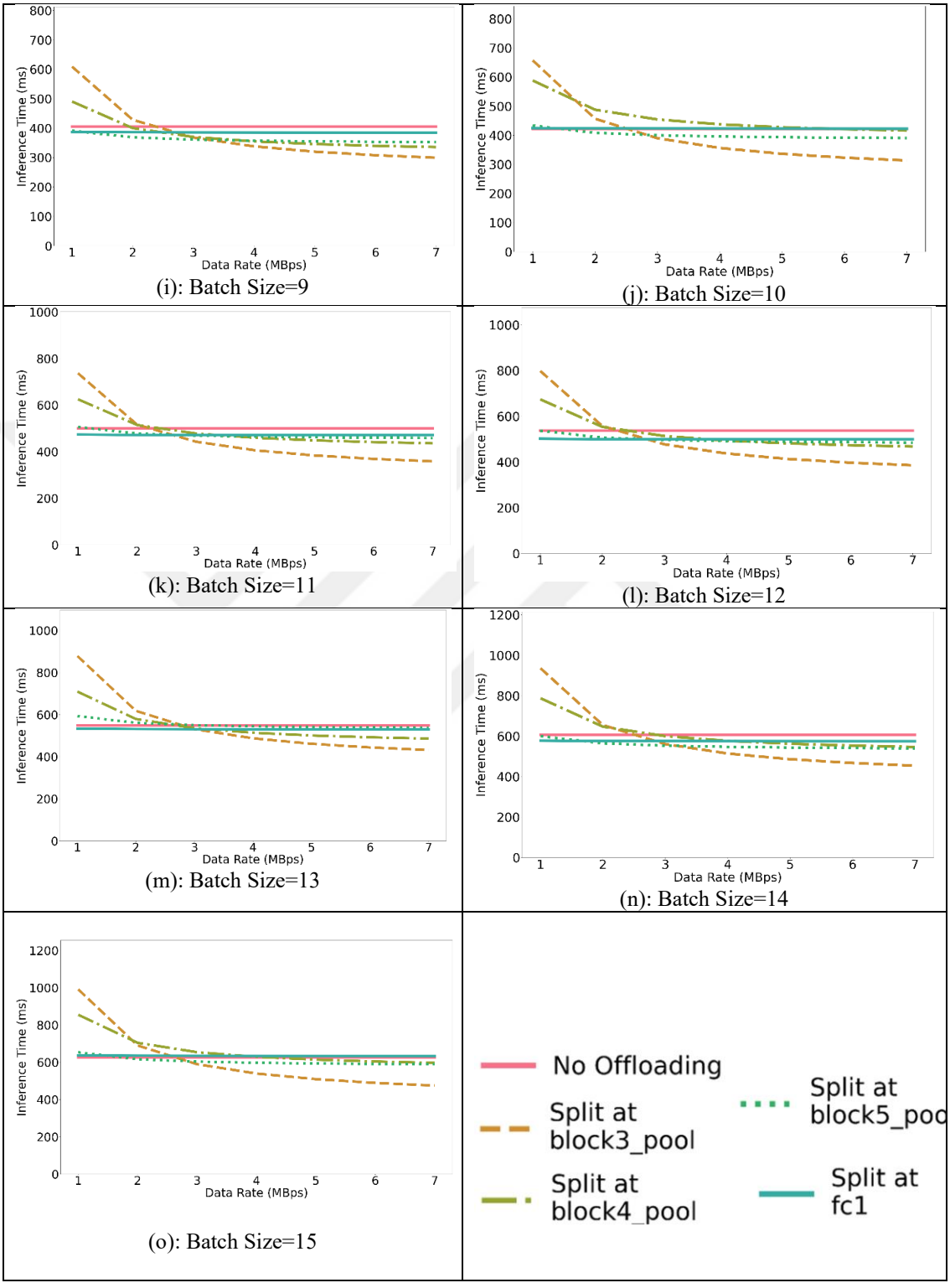


(a): Batch Size=1 (b): Batch Size=2

(c): Batch Size=3 (d): Batch Size=4

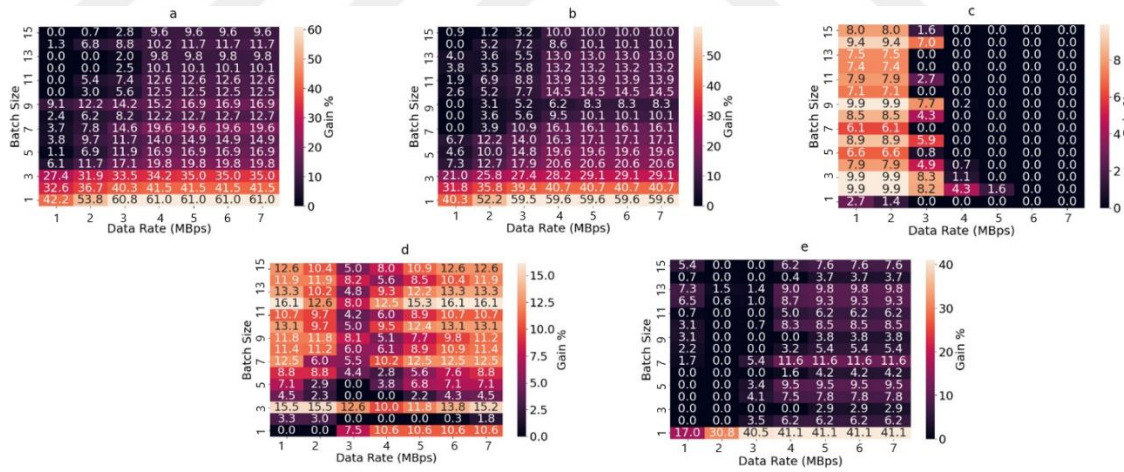
(e): Batch Size=5 (f): Batch Size=6

(g): Batch Size=7 (h): Batch Size=8



Şekil 4. Veri Hacimleri Arasında Çıkarım Süreleri - Batch Boyut 1'den 15'e (a-o).

Bu bağlamda, şekil 4'te gösterilen sonuçlar, artan batch boyutları ve azalan bant genişliğinin, aktarım süresindeki artış nedeniyle çıkarım sürelerini önemli ölçüde uzattığını ortaya koymaktadır. Elde edilen kazanç, bölme noktasının rastgele seçimine, çıkarım süresine, batch boyutuna ve ağ koşuluna göre dinamik olarak hesaplanması gerekmektedir. Dinamik bölme noktalarının rastgele seçime kıyasla sağladığı avantajlar nicel olarak değerlendirilmekte ve şekil 5'te sunulmaktadır. Sonuç olarak, dinamik bölme sürecinin rastgele kazanç durumu her batch boyutu ve bant genişliği ayarı için görselleştirilmektedir. Bu analiz, işlem yükünün devredilmediği senaryolarda işlem sürelerinde %61'e varan bir iyileşme sağlandığını ve özellikle ‘fc1’, ‘block3_pool’, ‘block4_pool’ ve ‘block5_pool’ katmanları dahil olmak üzere çeşitli katmanlarda önemli performans artışları gözlemlendiğini göstermektedir. Ayrıca, ağın farklı koşullar altındaki güvenilirliği, farklı batch boyutlarında tersine nicemleme sonrasında paket kayıplarını karşılaştırarak değerlendirilmekte ve artan veri hacimleri ile paket kayıpları arasında doğrudan bir ilişki olduğu doğrulanmaktadır.



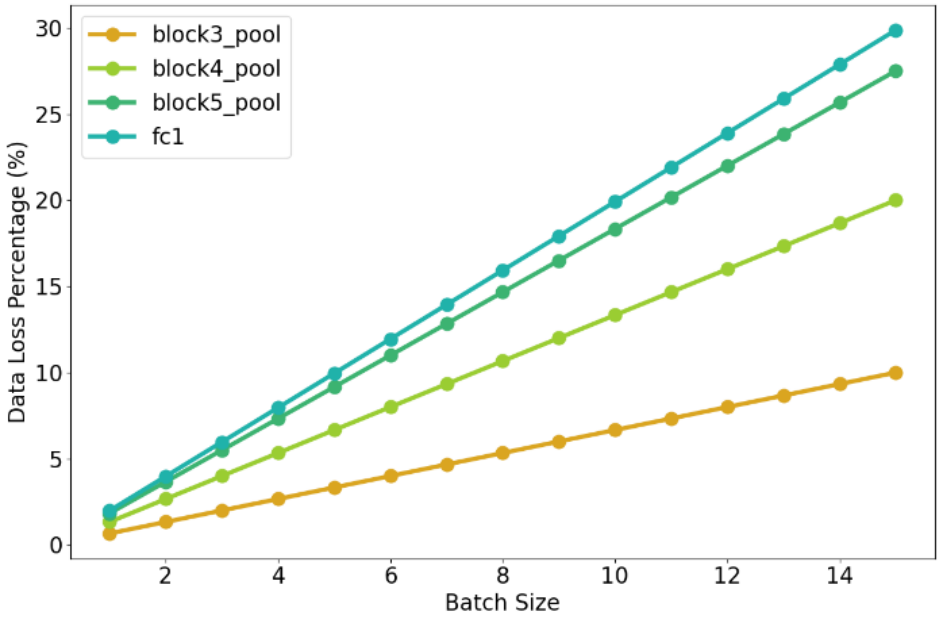
Şekil 5. Rastgele Bölünme Senaryolarında Farklı Katmanlar Arasında Dinamik Bölünme Noktanın Performans Karşılaştırması (a: ‘Boşaltma Yok’ b: ‘fc1’ katmanı, c: ‘block3_pool’ katmanı, d: ‘block4_pool’ katmanı, e: ‘block5_pool’ katmanı).

Deneysel bulguların tekrarlanabilirliğini ve genelleştirilebilirliğini garantilemek için araştırmamızda kullanılan donanım özelliklerinin kapsamlı bir hesabını sunulmaktadır. Çalışmalarda kullanılan uç cihaz, IoT ve gömülü sistemlerin düşük güçlü bir bilgi işlem birimi karakteristiğine örnek teşkil eden Intel i5-7300HQ CPU (2,50 GHz) ile donatılan dizüstü bilgisayar olmakta ve buluttaki işlem için üstün

hesaplama performansı sağlayan NVIDIA Tesla T4 GPU'ya sahip uzak bir sunucuda gerçekleşmektedir. Uç ve bulut cihazları arasındaki işleme yeteneklerindeki önemli farklılıkları göz önünde bulundurarak uç taraf yürütmeyi, bulut tarafı çıkarımını ve veri aktarım gecikmesini ayrı ayrı değerlendirerek çıkarım süresini metodik olarak değerlendirilmektedir. Şekil 4 ve 5, farklı donanım yeteneklerinin performans üzerindeki etkisini daha ayrıntılı olarak inceleyerek önerilen dinamik bölme noktası seçiminin gerçek zamanlı bilgi işlem ve ağ sınırlamalarına verimli bir şekilde uyum sağladığını göstermektedir. Bu durum, yöntemin çeşitli donanım yapılandırmalarında uygulanabilirliğini garanti ederek onu çeşitli IoT ve bulut tabanlı senaryolar için ideal hale getirmektedir.

2.2.3. Veri Yeniden Yapılandırma

İletim esnasında veri kaybını azaltmak amacıyla özellik haritalarından kaybolan veri paketlerini yeniden yapılandırabilen bir AE uygulanmaktadır. Bu AE, veri bütünlüğünü korumak için maksimum havuzlama (ing: max-pooling) ile aşağı örnekleme (ing: down-sampling) ve yukarı örnekleme (ing: up-sampling) içeren evrişimsel katmanlardan oluşmaktadır. Paket veri kaybı hem ağ koşulları hem de aktarılan veri miktarı nedeniyle aktarım sırasında değişiklik göstermektedir. Şekil 6, dinamik olarak seçilen bölme noktaları boyunca bu değişimleri göstermekte ve paket kayıplarının veri hacmi arttıkça yükseldiğini ortaya koymaktadır. Batch boyutu arttıkça ağ üzerinden daha fazla veri iletilmekte ve bu durum paket kaybı riskini artırmaktadır. Ayrıca, DNN mimarisi içinde özellikler son katmana yaklaştıkça daha karmaşık hale gelmekte ve bu durum yalnızca üretilen veri miktarını artırmakla kalmayıp özellikle daha derin katmanlarda paket kayıplarının daha sık yaşanmasına neden olmaktadır.



Şekil 6. Bulut tarafından katman çıktılarının alınması sırasında dinamik bölünme noktalarının ve ilişkili veri paketi kayıplarının analizi.

DSPCI-MTL metodolojisini, AE ile ve AE olmadan karşılaştırmalı performansı literatürde bulunan güncel yöntemlerle (Alvar & Bajic, 2019), (Alvar & Bajic, 2020), (Hu vd., 2024), (Capogrosso vd., 2023), (Y. Wang vd., 2021), (Ahuja vd., 2023), (Zhou vd., 2023), SiLRTC, HaLRTC ve CALTeC dâhil olmak üzere değerlendirilmektedir. SiLRTC, HaLRTC ve CALTeC yaklaşımları, AE'ye benzer şekilde eksik veri paketlerini yeniden yapılandırmak için önerilen metodolojiye entegre edilmekte ve etkileyici sonuçlar elde edilmektedir. Tablo 6'da vurgulandığı üzere, AE'nin kullanımı model doğruluğunu önemli ölçüde artırmakta ve özellikle segmentasyon, derinlik tahmini ve sınıflandırma görevlerindeki hata oranlarını azaltmaktadır. Bu bulgular, AE'nin veri güvenilirliğini artırmada ve karmaşık kentsel ortamlarda sistem performansını iyileştirmede etkili olduğunu ortaya koymaktadır. Bununla birlikte, önerilen yöntem, bilinen literatürdeki mevcut çalışmaların üç farklı görevi aynı anda ele almaması nedeniyle benzersiz bir performans sergilemektedir. Tablo 6'da sunulan segmentasyon, derinlik tahmini ve sınıflandırma sonuçlarının detaylı incelemesinde bulunan yöntemlerin temel yöntemlere kıyasla daha iyi performans gösterdiğini ancak SiLRTC, HaLRTC ve CALTeC'in yeniden yapılandırma uygulanmayan DSPCI-MTL'e kıyasla daha düşük etkinlik sergilediğini göstermektedir. Özellikle AE ile entegre edilen DSPCI-MTL'in üstün performans

göstermesi, tahmin doğruluğunu ve sistemin dayanıklılığını artırma potansiyelini daha da vurgulamaktadır.

Tablo 6. Segmentasyon (mIoU %) ve Derinlik Tahmini (RMSE) ve Sınıflandırma (Etiket Doğruluğu) Görevlerine Dayalı Yöntemlerin Karşılaştırılması.

Çalışma	Segmentasyon (mIoU %)	Derinlik Tahmini (RMSE)	Sınıflandırma (Sınıf Doğruluğu %)
(Alvar & Bajic, 2019)	57.00	8.26	-
(Alvar & Bajic, 2020)	62.68	7.86	-
(Hu vd., 2024)	59.70	-	-
(Y. Wang vd., 2021)	60.58	-	-
(Capogrosso vd., 2023)	-	-	78.00
(Ahuja vd., 2023)	-	-	74.91
(Zhou vd., 2023)	-	-	69.54
DSPCI-MTL (Yeniden Yapılandırmasız)	60.53	13.72	48.78
DSPCI-MTL ve SiLRTC	41.36	16.93	63.20
DSPCI-MTL ve HaLRTC	44.13	15.07	62.68
DSPCI-MTL ve CALTeC	49.04	13.02	72.71
DSPCI-MTL ve AE (Önerilen)	**65.48**	**5.84**	**83.65**

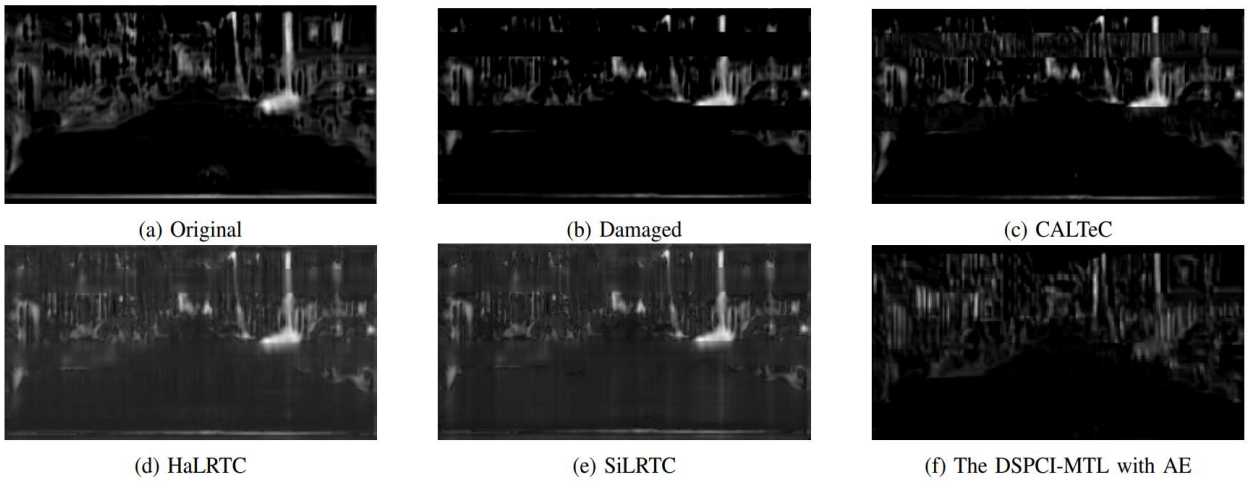
Daha önce belirtildiği üzere, veri yeniden yapılandırma başarısı SSIM, PSNR, MAE ve DSSIM metrikleri kullanılarak değerlendirilmektedir. Bu kapsamda tablo 7, PSNR, SSIM, MAE ve DSSIM metriklerine dayalı olarak çeşitli DSPCI-MTL yeniden yapılandırma yöntemlerinin karşılaştırmalı değerlendirmesini sunmaktadır. Sonuçlar, önerilen AE tabanlı yaklaşımın sırasıyla 32.37 ve 0.851 olan en yüksek PSNR ve SSIM değerlerine ulaştığını ve SiLRTC, HaLRTC ve CALTeC yöntemlerine kıyasla üstün yeniden yapılandırma doğruluğu sağladığını göstermektedir. Ayrıca, AE tabanlı yöntem, 0.018 olan en düşük MAE ve 0.149 olan DSSIM puanlarını elde ederek, yapısal tutarlılığı koruma ve yeniden yapılandırma hatalarını en aza indirme konusundaki etkinliğini bir kez daha vurgulamaktadır. Bu bulgular, AE tarafından sağlanan özellik haritası restorasyonundaki önemli iyileşmeyi ortaya koymakta ve yeniden yapılandırılan ve orijinal özellikler arasındaki benzerliği artırmaktadır. Sonuç olarak, önerilen yaklaşım, CI çerçevelerinde DNN tahmin doğruluğunu ve genel sistem performansını önemli ölçüde artırmaktadır.

Tablo 7. Veri Yeniden Oluşturmaya Dayalı Yöntemlerin Karşılaştırılması (PSNR, SSIM, MAE ve DSSIM).

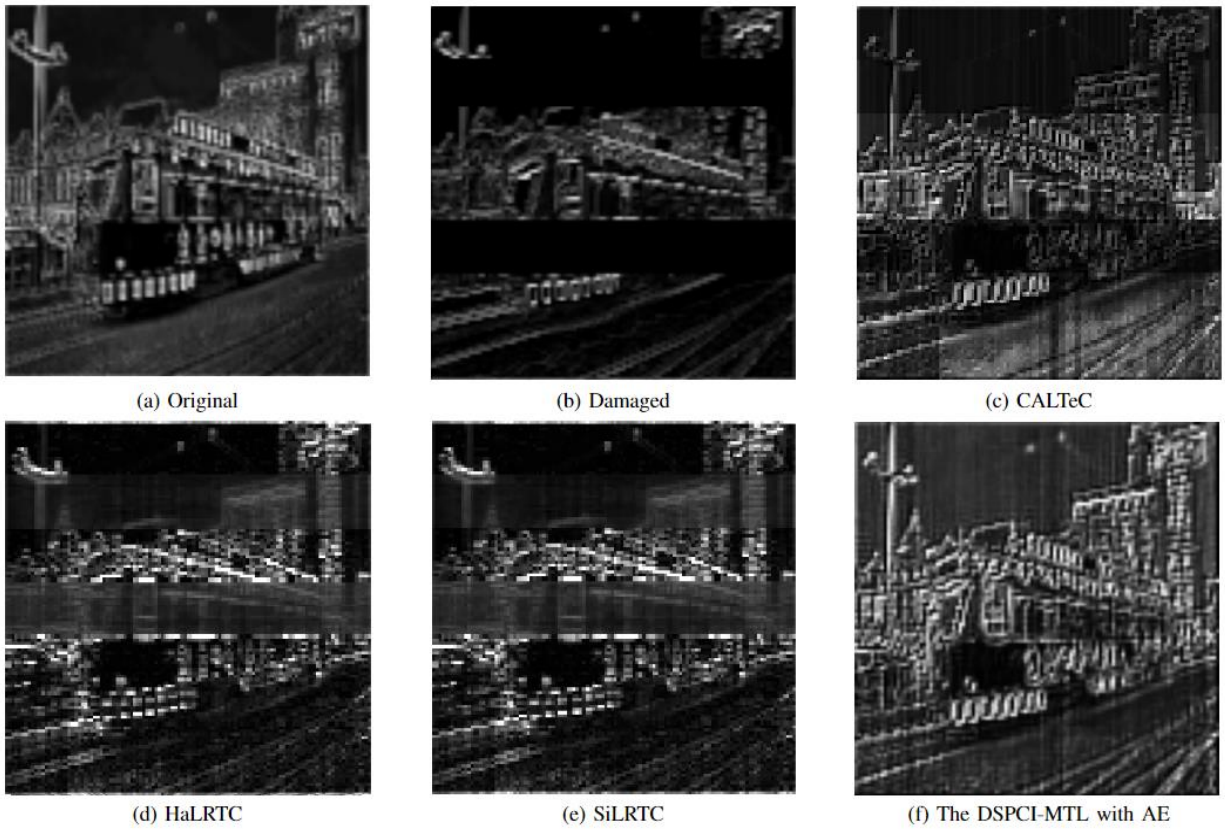
Metod	PSNR	SSIM	MAE	DSSIM
DSPCI-MTL ve SiLRTC	16.82	0.115	0.067	0.885
DSPCI-MTL ve HaLRTC	17.49	0.135	0.059	0.865
DSPCI-MTL ve CALTeC	26.34	0.229	0.043	0.771
DSPCI-MTL ve AE (Önerilen)	**32.37**	**0.851**	**0.018**	**0.149**

Ayrıca, şekil 7 ve 8, sırasıyla Cityscapes ve ImageNet veri setleri için önerilen DSPCI-MTL'in ayrıntılı görsel bir gösterimini sunarak AE'nin üstün yeniden yapılandırma yeteneklerini sergilemektedir. Böylece, yöntemin daha karmaşık ve büyük görüntü veri setlerini işleme konusundaki etkinliğini vurgulamakta ve AE'nin kaybolan görüntü paketlerini doğru bir şekilde yeniden yapılandırma ve yüksek kaliteli çıkarım sağlama yeteneğini ortaya koymaktadır. Bu durum, daha zorlu ve çeşitli veriler bağlamında da geçerli olmaktadır. Sunulan görsel kanıtlar, AE ile DSPCI-MTL'in, gerçek dünya uygulamalarında kullanım potansiyelini güçlendirmekte ve veri kurtarma konusundaki dayanıklılık ve doğruluk kritik öneme sahip olmaktadır. Ayrıca bu şekil, daha önce belirtilen yöntemler kullanılarak hasar görmüş bir katmanlı tensör kanalının restorasyon çıktısını göstermektedir. CALTeC, HaLRTC ve SiLRTC yöntemleri kaybolan verileri etkin bir şekilde geri kazandırırken, önerilen DSPCI-MTL ve AE yöntemi olağanüstü performans sergileyerek veriyi doğru bir şekilde geri yüklemede en başarılı olan yöntem olarak öne çıkmaktadır. Bu sonuç, AE'nin dayanıklılığını ve pratik koşullarda veri kaybını ele almadaki etkinliğini doğrulamakta ve veri yeniden yapılandırmasında yüksek doğruluğun kritik olduğu senaryolar için özellikle uygun olduğunu göstermektedir. Deneyler sırasında, AE ile yeniden yapılandırma işleminin ortalama 20 saniyelik bir eğitim gecikmesi oluşturduğu gözlemlenmektedir. Bu gecikmenin etkisini azaltmak için sistemimiz, iki farklı senaryo altında çalışacak şekilde uyarlanmıştır. İlk senaryo, anlık çıkarımın kritik olduğu durumları ele almaktadır. Bu durumda, AE işlemi uygulanmadan hemen sonuç gerektiren görevlerin verimli ve gecikmesiz bir şekilde işlenmesi sağlanmaktadır. Bu yapılandırma, gerçek zamanlı izleme sistemleri veya acil durum müdahale

uygulamalarında olan zamanın çok önemli olduğu, kararların hızlı bir şekilde alınması gereken ve veri gecikmesinin sonuçları önemli ölçüde etkileyebileceği uygulamalarda özellikle kritik öneme sahip olmaktadır. İkinci senaryoda ise verilerin buluta hızlı bir şekilde aktarılması ancak anında analiz edilmesinin gerekmediği durumlarda olan veri işleme için acil bir gerekliliğin olmadığı durumlarda uygulanabilmektedir. Bu durumda, sistem DSPCI-MTL metodolojisinin tüm yeteneklerinden faydalanarak verilerin buluta verimli bir şekilde iletilmesini ve AE'nin özellik haritası yeniden yapılandırmasını sağlamaktadır. Böylece, DNN çıktıların doğruluğunu ve güvenilirliğini artırarak AE'nin veri kalitesini geliştirme yeteneğinden yararlanmakta ve bu durum daha uzun işlem süresi pahasına yapılmaktadır. Bu yaklaşım, ayrıntılı veri analizi veya toplu işleme senaryoları için anında elde edilen sonuçlardan ziyade verilerin bütünlüğünün ve hassasiyetinin daha kritik olduğu uygulamalar için ideal olmaktadır.



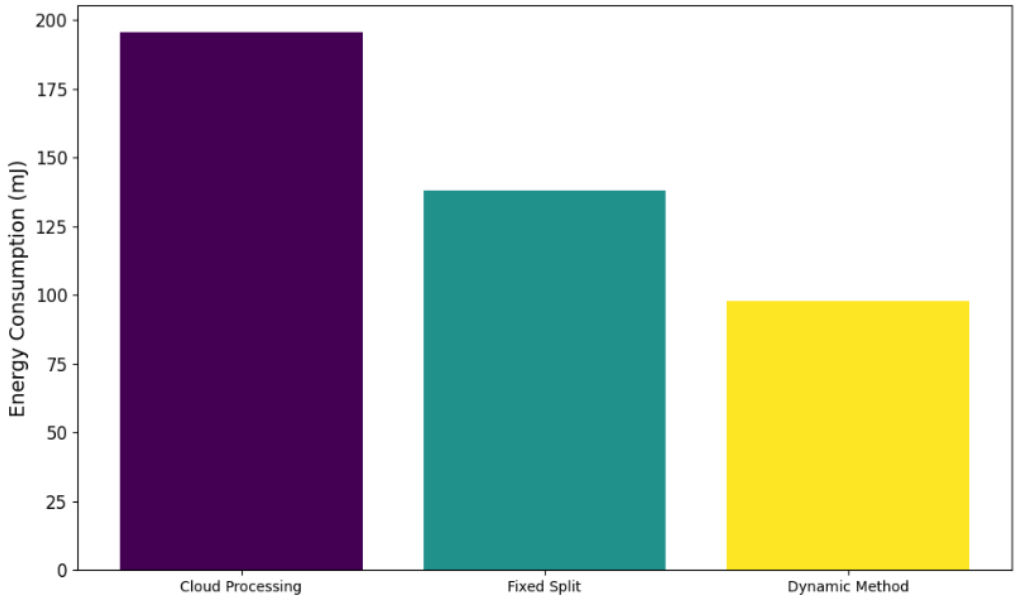
Şekil 7. Cityscapes veri setine dayalı sonuçlar. (a) Orijinal; (b) Hasarlı; (c) CALTeC, (d) HaLRTC, (e) SiLRTC, (f) DSPCI-MTL, AE ile onarıldı.



Şekil 8. ImageNet veri setine dayalı sonuçlar. (a) Orijinal; (b) Hasarlı; (c) CALTeC, (d) HaLRTC, (e) SiLRTC, (f) DSPCI-MTL, AE ile onarıldı.

2.2.4. Enerji Tüketimi Analizi

Çıkarım sürelerini iyileştirmenin yanı sıra enerji tüketiminin incelenmesi de bu kısımda ele alınmaktadır. CI çerçevesi, kaynak kullanımını iyileştirmeyi ve enerji verimliliğinin değerlendirilmesini kapsamlı bir performans incelemesi için gerekli kılmaktadır. Bu sebeple, Şekil 9'da önerilen DSPCI-MTL yönteminin geleneksel yöntemlerle karşılaştırıldığında enerji kullanımını göstermektedir. Sonuç olarak, tüm işlemler bulutta yürütüldüğünde enerji tüketimi 195,543 mJ iken, sabit bölme noktası tekniği 138,04 mJ tüketmekte ve önerilen dinamik yöntem 97,771 mJ tüketmektedir. Sonuçlar, önerilen stratejinin enerji kullanımını belirgin şekilde azalttığını ve geleneksel yaklaşımlardan daha verimli bir çerçeve sağladığını göstermektedir. Sabit bölme noktası tekniğine göre %29,2'nin üzerinde enerji tasarrufu elde etmekte ve tamamen bulut tabanlı işleme modeline kıyasla %50 tasarruf sağlamaktadır.



Şekil 9. Farklı İşleme Yöntemlerinin Enerji Tüketimi Karşılaştırması.

2.2.5. Farklı Ağ Gecikme ve Veri Hacimleri Koşullarında Performans Analizi

Önerilen tekniğin farklı ağ gecikmesi durumlarında performansı Tablo 8'de sunulmaktadır. Önerilen dinamik bölmenin çıkarım süresi açısından sabit bölme yönteminden daha verimli performans göstermektedir. Uygulanan simülasyonda, düşük gecikme ayarlarında (10 ms gecikme), dinamik bölme yaklaşımı çıkarım süresini %47,5 oranında azaltarak 167,9 ms'lik bir işlem süresi sağlarken sabit bölme yöntemi için bu süre 320 ms olmaktadır. Orta gecikme koşullarında (50 ms gecikme), dinamik bölme yaklaşımı çıkarım süresini 240 ms'ye düşürerek sabit bölme yönteminin 420 ms'sine kıyasla %42,9'luk bir iyileştirme sağlamaktadır. Yüksek gecikme durumlarında bile (100 ms gecikme), dinamik bölme yöntemi önemli bir fayda sağlayarak sabit bölme stratejisi için 610 ms'ye kıyasla 350 ms'lik bir çıkarım süresi elde ederek %42,6'lık bir azalma göstermektedir. Bu sonuçlar, algoritmanın dalgalanan ağ koşullarına sahip gerçek zamanlı uygulamalarda uyarlanabilirliğini ve sağlamlığını doğrulayarak uç bulut dağıtılmış sistemlerde optimize edilmiş işlem verimliliği sağlamaktadır.

Tablo 8. Farklı ağ gecikme koşulları altında dinamik bölme noktası seçiminin performansı.

Ağ Gecikmesi	Sabit Bölünme (Çıkarım Süresi ms)	Dinamik Bölünme (Çıkarım Süresi ms)
Düşük (10 ms)	320	167.9
Orta (50 ms)	420	240
Yüksek (100 ms)	610	350

Ayrıca, tablo 9, sabit ve dinamik bölme yapılandırmaları için farklı veri hacimlerinin çıkarım süresi üzerindeki etkisini göstermektedir. Bulgular, önerilen dinamik bölme metodolojisinin artan veri hacimlerine etkili bir şekilde uyum sağladığını ve küçük batch boyutları için %76,3'e kadar dikkate değer bir verimlilik artışı sağladığını göstermektedir. Ek olarak daha büyük batch boyutları için önemli performans artışları göstermekte ve sırasıyla 5, 10 ve 15 parti boyutları için %42,3, %38,3 ve %35,4'lük iyileştirmeler sağlamaktadır. Sonuçlar, gerçek zamanlı dağıtılmış işlemede dinamik bölme seçimi yaklaşımının ölçeklenebilirliğini ve hesaplama verimliliğini vurgulamaktadır. Dahası, deneysel bulgularımız, önerilen dinamik bölme noktası seçimi algoritmasının ağ gecikmelerindeki ve veri hacimlerindeki dalgalanmalara etkili bir şekilde uyum sağladığını doğrulamaktadır. Sabit bölme stratejisiyle karşılaştırıldığında, dinamik seçim yöntemi, özellikle yüksek gecikmeli ortamlarda ve kapsamlı veri işleme durumlarında çıkarım süresini önemli ölçüde azaltmaktadır. Bu teknik, düşük gecikmeli senaryolarda maksimum %47,5'lik bir iyileştirme ve küçük batch boyutlarında %76,3'lük bir verimlilik artışı elde ederek çeşitli ağ koşullarında dayanıklı performansı garanti etmektedir. Bu bulgular, IoT tabanlı akıllı sistemler ve uç bulut işbirlikçi zeka çerçeveleri dahil olmak üzere gerçek zamanlı uyarlanabilir bilgi işlem ortamları için metodolojimizin uygunluğunu doğrulamaktadır.

Tablo 9. Artan Veri Hacminin Dinamik Seçim Süresine Etkisi.

Batch size	Sabit Bölünme (ms)	Dinamik Bölünme (ms)	İyileştirme (%)
1	86.6	20.5	76.3%
5	215.4	124.3	42.3%
10	409.1	252.1	38.3%
15	588.5	380.3	35.4%

2.2.6. Ek Yük Analizi

Önerilen dinamik bölmenin AE tabanlı yeniden yapılandırma ile performans etkisini değerlendirmek için tablo 10, hem sabit bölme hem de dinamik bölme ayarları altında farklı işleme bileşenleri arasında gecikmenin ayrıntılı bir dökümünü sağlamaktadır. AE, 60 ms'lik ek bir işleme gecikmesi yaşamaktadır. Ancak bu durum, uç işleme, özellik iletimi ve bulut işleme aşamalarındaki geliştirmelerle hafifletilmektedir. Uç tarafı bilgi işlem, özellik karmaşıklığının azalmasından yararlanmakta ve bu durum işleme süresinde 20 ms'lik bir azalmaya yol açmaktadır. Dahası, sıkıştırılmış özellik haritaları daha az bant genişliği gerektirdiğinden özellik iletim gecikmesi 10 ms azaltmakta ve bu durum daha hızlı veri aktarımı sağlamaktadır. Bulut işleme gecikmesi, daha verimli çıkarım hesaplamalarını kolaylaştıran iyileştirilmiş özellik gösterimi nedeniyle 20 ms azaltmaktadır. Sonuç olarak, genel gecikme 350 ms'den 360 ms'ye hafifçe artmakta ve ek AE yeniden yapılandırma yükünü yansıtmaktadır. Buna rağmen, özellik yeniden yapılandırma ve model sağlamlığı önemli ölçüde iyileştirmektedir. Araştırmamız, AE modülünün bir hesaplama yükü getirmesine rağmen, genel çıkarım gecikmesini yalnızca marjinal olarak etkilediğini doğrulamaktadır. AE işleme ve sistem genelindeki optimizasyonlar arasındaki denge, önerilen DSPCI-MTL çerçevesinin gerçek zamanlı verimliliği korurken veri bütünlüğünü ve iletim kayıplarına karşı sağlamlığını artırmasını sağlamaktadır. Gelecekteki araştırmalar, hesaplama maliyetlerini daha da azaltmak ve kaynak kısıtlı ortamlarda dağıtım verimliliğini artırmak için hafif oto kodlayıcı tasarımlarını, nicemleme tekniklerini veya bilgi damıtma yöntemlerini inceleyebilir.

Tablo 10. Gecikme Bileşenlerinin Karşılaştırılması (ms).

İşleme Bileşeni	Sabit Bölünme Gecikmesi	Dinamik Bölünme + AE Gecikmesi	AE Nedeniyle Genel Gecikme
Uç İşlem	150	130	-20
Özellik İletimi	80	70	-10
Bulut İşleme	120	100	-20
AE Veri Yeniden Yapılandırma	0	60	+60
Toplam Gecikme	**350**	**360**	**+10**

2.3. DSPCI-MTL DEĞERLENDİRMESİ

2.3.1. Tartışma

IoT cihazlarında DNN uygulamaları, sınırlı donanım kapasitesi, kısıtlı bant genişliği, uzun veri aktarım süreleri ve iletim hatalarına bağlı görüntü paketi kayıpları olmak üzere önemli zorluklar içermektedir. Bu sorunlara çözüm olarak DSPCI-MTL yöntemi önerilmektedir. Önerilen yöntem, üç farklı DNN görev alanında eğitilerek E2E-E2C cihazlarında uygulanmaktadır. 1–7 MBps bant genişliği ve 1 ila 15 arasında değişen yığın boyutları altında gerçekleştirilen performans değerlendirmeleri, bireysel ağ katmanlarının işlem verimliliğinin ağ koşullarına bağlı olarak önemli ölçüde değiştiğini ortaya koymaktadır. Özellikle, yığın boyutu üçten büyük olduğunda ve bant genişliği sınırlı olduğunda, 'block3_pool' katmanı en kısa çıkarım süresine ulaşmaktadır. Öte yandan, daha yüksek bant genişliği koşullarında bu katman üstün işlem hızlarına erişmektedir. Aynı zamanda, 'block4_pool' ve 'block5_pool' katmanları düşük bant genişliğinde daha iyi çıkarım süreleri sergilemektedir. Çalışmanın önemli bulgularından biri ise ağın 'block3_pool' katmanında bölünmesinin, tüm işlemlerin yalnızca uç cihazda gerçekleştirildiği yapılandırmalara kıyasla toplam işlem süresini %38'e kadar azaltmaktadır. Ancak, bölme katmanının seçimi, çıkarım süresi, yığın boyutu, ağ koşulları ve çeşitli etkenlere göre dinamik olarak optimize edilmesi gerekmektedir. Sonuçlar, farklı yığın boyutları ve bant genişliği ayarlarında yapılan bölme işlemlerinin sağladığı dinamik kazancı da göstermektedir. Bu analiz, offloading yapılmayan senaryolarda işlem süresinde %61'e varan iyileşmeler sağlandığını ve 'fc1', 'block3_pool', 'block4_pool' ve 'block5_pool' gibi katmanlarda anlamlı performans artışları elde edildiğini ortaya koymaktadır. Buna ek olarak, uygulanan AE yöntemi, iletim sırasında öznitelik haritası içindeki benzerliklere dayalı olarak veriyi yeniden yapılandırmak üzere tasarlanmakta ve bu yönüyle geleneksel yöntemlerden ayrılmaktadır. AE yöntemi, geleneksel CALTeC, HaLRTC ve SiLRTC yöntemlerinden üstün performans sergileyerek 32.37 PSNR ve 0.851 SSIM değerlerine ulaşmaktadır. Bu yenilikçi yaklaşım, değerlendirilen üç görevde de kaybolan paketleri etkili bir şekilde yeniden yapılandırarak DNN tahmin doğruluğunu artırmaktadır. Enerji tüketimi analizinde ise, önerilen DSPCI-MTL

yönteminin geleneksel yaklaşımlara kıyasla enerji tüketimini önemli ölçüde azalttığını göstermektedir. Dinamik bölme noktası seçimi mekanizması, hesaplamayı uç ve bulut cihazları arasında dengeli bir şekilde dağıtarak enerji harcamalarının iyileştirmektedir. Sonuçlar, bu yöntemin tamamen bulut tabanlı işleme kıyasla %50, sabit bölme noktası kullanılan yöntemlere kıyasla ise %29.2 oranında enerji tasarrufu sağladığını doğrulamaktadır. Bu da yüksek performans korunurken enerji verimliliğinin sağlandığını göstermektedir. Bir diğer önemli değerlendirme kriteri ise yöntemin farklı ağ gecikmesi koşulları ve veri hacmi altında gösterdiği performans olmaktadır. Deneysel sonuçlar, dinamik bölme yönteminin değişen gecikme koşullarında sabit bölme yöntemine kıyasla çıkarım süresi açısından üstünlük sağladığını göstermektedir. Yüksek gecikmeli senaryolarda (100 ms gecikme), önerilen yöntem çıkarım süresini sabit bölme stratejisine göre %42.6 oranında azaltmaktadır. Benzer şekilde, veri hacminin artması çıkarım süresini olumsuz etkilerken, dinamik bölme yöntemi özellikle küçük yığın boyutlarında %76.3'e varan iyileştirme sunmakta ve büyük yığın boyutlarında da önemli kazançlar sağlamaya devam etmektedir. Ek yük analizine bakıldığında, AE entegrasyonu sistemde 60 ms'lik ek bir işlem gecikmesine neden olmakta ve toplam gecikmeyi 360 ms'ye çıkarmaktadır. Ancak bu artış, uç işlem, öznitelik iletimi ve bulut tarafı işlemlerinde sağlanan iyileştirmelerle dengelenmektedir. Özellikle, uç taraflı işlem süresi 20 ms azaltılırken, öznitelik haritalarının sıkıştırılması sayesinde öznitelik iletimi 10 ms hızlanmaktadır. Aynı şekilde, bulut tarafı işlemde de öznitelik temsili iyileştirildiği için 20 ms'lik bir azalma sağlanmaktadır. Genel gecikmede küçük bir artış olsa da, bu optimizasyonlar AE'nin sistem dayanıklılığını ve veri bütünlüğünü artırmasını sağlamakta ve gerçek zamanlı verimliliği olumsuz etkilememektedir. Gelecek çalışmalar, hesaplama verimliliğini daha da artırmak için hafif autoencoder mimarileri veya bilgi distilasyonu tekniklerinin uygulanmasını araştırabilir. Yığın boyutunun sistem performansına etkisi, dikkatli değerlendirilmesi gereken kritik bir faktör olmaktadır. Küçük yığın boyutları (örneğin yığın boyutu = 1) daha düşük çıkarım gecikmesi sağlasa da model doğruluğunda kayıp oluşmasına yol açmaktadır. Deneysel bulgular, yığın boyutu 10'un doğruluk, hesaplama verimliliği ve kaynak kullanımı arasında en uygun dengeyi sunduğunu göstermektedir. Bu eşik değerinin üzerindeki yığın boyutlarında doğruluk artışı sınırlı kalırken, hesaplama yükü önemli

ölçüde artmaktadır. Performansın daha da optimize edilmesi için, adaptif yığın boyutlandırma mekanizmalarının uygulanması önerilebilir. Böylece gerçek zamanlı ağ koşulları ve sistem yüküne göre dinamik ayarlamalar yapılabilir. Son olarak, sağlamlık analizi, önerilen yöntemin paket kaybına karşı dayanıklılığını ortaya koymaktadır. Ağda paket kaybı arttıkça model performansında kayıp oluşturmaktadır. AE ile entegre DSPCI-MTL yöntemi %10–30 arası kayıplara karşı güçlü direnç göstermektedir. Ancak %50'nin üzerindeki kayıplarda, özellikle COCO ve ADE20K gibi karmaşık veri kümelerinde doğrulukta ciddi kayıplar yaşanmaktadır. Bu bulgular, yüksek derecede kararsız ağ ortamlarında sağlamlığı artırmak amacıyla federe öğrenme ya da hibrit yeniden yapılandırma yöntemleri gibi ek hata düzeltme stratejilerinin gerekliliğini vurgulamaktadır.

2.3.2. Sonuç

Bu çalışma, IoT ortamlarında hesaplama süreçlerini iyileştirmek amacıyla CI yaklaşımlarını etkin bir şekilde kullanan yenilikçi bir yöntem olan DSPCI-MTL metodolojisini tanıtmaktadır. Bu yaklaşım, kenar ve bulut bilişim kaynaklarının verimli kullanımını sağlayarak çok görevli öğrenme tekniğini kullanmakta ve görev yürütme sürecini optimize etmektedir. Gerçek zamanlı veriler ve ağ koşullarına dayalı olarak optimal ayrım noktalarının dinamik biçimde belirlenmesi sayesinde işlem sürelerinde kayda değer iyileşmeler sağlanmakta ve hesaplama yükü azaltılmaktadır. Ayrıca, iletim sırasında kaybolan verilerin yeniden yapılandırılması amacıyla AE entegrasyonu, değişken ağ senaryoları altında veri bütünlüğünün korunmasında ve sistem performansının artırılmasında yöntemin etkinliğini pekiştirmektedir. Genel olarak, DSPCI-MTL metodolojisi, IoT sistemlerinde karşılaşılan karmaşık hesaplama zorluklarının üstesinden gelmek amacıyla MTL ile CI entegrasyonunun potansiyelini vurgulamakta ve derin öğrenme teknolojilerinin pratik uygulamaları için yeni bir referans noktası oluşturmaktadır. Verinin etkin işlenmesi ve yönetimini güvence altına alarak, kent analitiğinden ileri düzey IoT sistemlerine kadar pek çok uygulama alanında veri bütünlüğü, işlem verimliliği ve zamanında yürütme olmak üzere çeşitli kritik gereksinimlerin karşılanmasında önemli bir ilerleme vaat etmektedir. DSPCI-MTL tekniği önemli gelişmeler gösterse

de geliştirmeye açık yönler de ortaya koymaktadır. Gelecek çalışmalar için öne çıkan bazı araştırma yönleri şunlardır:

- Sistemin ultra düşük bant genişliği ya da ciddi ağ kararsızlığı gibi koşullara gerçek zamanlı adaptasyon yeteneğinin artırılması, otonom araçlar ve akıllı şehirler gibi kritik uygulamalar açısından yöntemin uygulanabilirliğini büyük ölçüde artıracaktır.
- DNN'in karmaşık yapısı, optimal ayrım noktasının belirlenmesinde çıkarım süresini olumsuz etkileyebilmektedir. Bu sorunun giderilmesi ve verimliliğin artırılması için meta-sezgisel yaklaşımlar kullanılabilir.
- Özellikle yüksek veri kaybı veya karmaşık görsel veri kümeleri içeren senaryolarda, Generative Adversarial Networks (GAN) gibi daha ileri düzey veri yeniden yapılandırma yöntemlerinin ya da hibrit modellerin araştırılması, görüntü kurtarma dayanıklılığını artırabilir.
- Gerçek zamanlı video analitiği ya da çevresel izleme gibi daha geniş bir IoT uygulama yelpazesini kapsayacak şekilde metodolojinin uyarlanması, etkinliğini ve ölçeklenebilirliğini artıracaktır.
- Donanımsal kaynakları sınırlı kenar cihazlarda hesaplama yükünü azaltmak için hafif modellerin ve sıkıştırma tekniklerinin geliştirilmesi, sınırlı donanım koşullarında bile sorunsuz işlevselliği güvence altına alacaktır.
- Kenar-Bulut etkileşimleri sırasında güvenli ve gizli veri yönetimini sağlamak amacıyla, mimariye şifreleme teknikleri ya da federe öğrenme yaklaşımlarının dahil edilmesi, gizlilik sorunlarını önemli ölçüde hafifletecektir.

ÜÇÜNCÜ BÖLÜM

3. Önerilen MH-tabanlı Metot

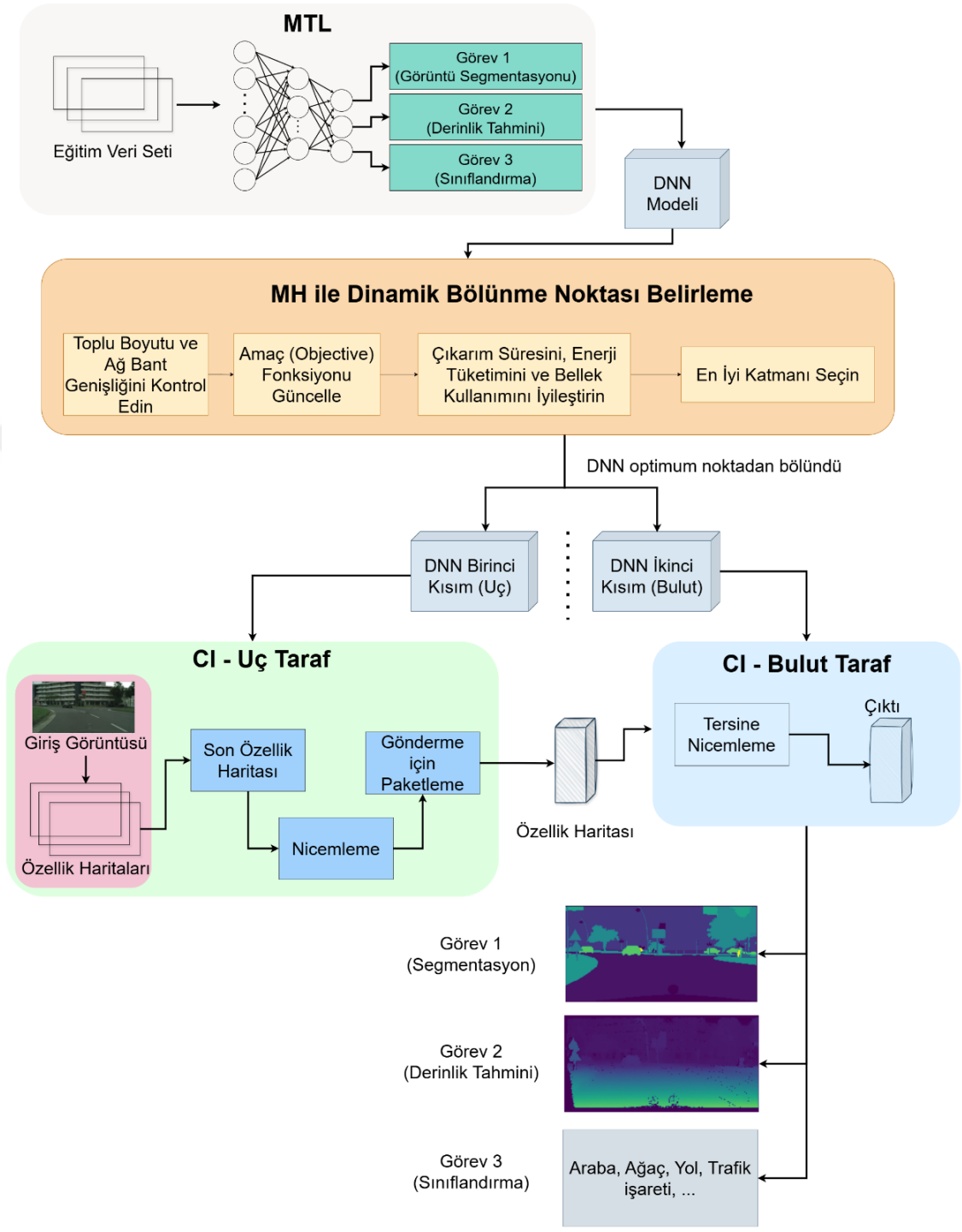
Bu bölümde, önerilen DSPCI-MTL'de kullanılan dinamik bölünme noktası yönteminin enerji tüketimi, hesaplama maliyeti ve bellek kullanımı açısından zorluklar ortaya çıkarması sebebiyle önerilen DSPCI-MH metodolojisi ayrıntılı olarak sunularak gerçekleştirilen deneyler doğrultusunda elde edilen bulgular değerlendirilmektedir. Öncelikle, geliştirilen yöntemin temel bileşenleri, teorik altyapısı ve uygulama aşamaları ele alınmakta ve kapsamlı deneysel analizler ışığında yöntemin etkinliği ve performansı tartışılmaktadır. Elde edilen sonuçlar, yöntemin IoT sistemlerinde enerji harcama, bellek kullanımı ve yürütme ve çıkarım sürelerini optimize etme konusundaki katkılarını ortaya koymaktadır.

3.1. ÖNERİLEN DSPCI-MH METODOLOJİSİ

Bu bölüm, IoT sistemlerindeki E2E-E2C cihazları için DNN uygulamalarında enerji ve bellek maliyetlerini optimize ederken çıkarımı hızlandırmak için geliştirilen metodolojiyi tanıtmaktadır. Önerilen yaklaşım Şekil 10'da sunulmaktadır. IoT cihazlarının belirli özelliklerine odaklanmak yerine, metodoloji verimli bulut tabanlı analizi kolaylaştırmak için bu cihazlar ile bulut arasındaki etkileşimi vurgulamaktadır. Bu nedenle, cihaza özgü kısıtlamalardan bağımsız genelleştirilmiş bir yöntem önerilmektedir. CI uygulamalarında, önerilen DSPCI-MH yöntemi, verimli DNN bölümlemesini etkinleştirmek için MTL'i CI ile entegre ederek bölme noktasını dinamik olarak belirlemektedir. Bir DNN'yi bölümlendirmek ve bileşenlerini farklı cihazlarda yürütmek, hesaplama yükünü dengelemek ve kaynak kullanımını optimize etmek için yaygın olarak benimsenen bir yaklaşım olmaktadır. Bu yöntemle, uç cihazlar ham görüntü verileri yerine çıkarılan özellik haritalarını iletmekte ve böylece bant genişliği tüketimini azaltırken veri gizliliğini artırmaktadır.

Önerilen yaklaşım, metasezgisel algoritmalardan yararlanarak optimum DNN bölme noktasını belirlemeyi amaçlamaktadır. Geleneksel yöntemlerde, bölümleme

noktası genellikle önceden tanımlanmış katman bazlı analizlere göre seçilmektedir. Ancak bu çalışma, bant genişliği ve toplu boyut değişikliklerine göre ayarlanan dinamik ve uyarlanabilir bir strateji sunmaktadır. Sonuç olarak, bellek kullanımı, enerji tüketimi ve çıkarım süresi açısından en verimli bölme noktası belirlenmekte ve bu da DNN modellerinin birden fazla cihazda etkili bir şekilde yürütülmesini sağlamaktadır.



Şekil 10. Önerilen DSPCI-MH Metodolojisinin İllüstrasyonu.

3.1.1. MH ile Optimum Bölünme Noktası Belirlenmesi

Bu bölüm, önerilen metodolojinin temel bileşenlerinden biri olan optimal katman bölme sürecini ele almaktadır. Optimum bölme noktasını belirlemek, modelin genel performansını ve hesaplama verimliliğini doğrudan etkileyen kritik bir

adım olmaktadır. Bu süreç, seçilen algoritmaları uygulayarak ve uygun bir amaç fonksiyon tanımlayarak gerçekleştirilmektedir.

3.1.1.1. Amaç Fonksiyonu

Geleneksel yöntemler bölme noktasını rastgele, matematiksel veya sezgisel olarak belirlerken, bu çalışma MH algoritmalarını kullanarak en iyi bölme noktasını dinamik olarak optimize etmektedir. Bu problem, denklem 9'da sunulan amaç fonksiyonunun en aza indirilmesi olarak modellenmektedir.

$$s_{Opt} = \underset{z \in \{0,L\}}{\operatorname{argmin}}(IT_z + w_1 M_z + w_2 E_z) \tag{9}$$

'Z' katmanında hesaplanan toplam çıkarım süresi IT_z olarak gösterilmektedir. Bu süre, ilgili katmandaki hesaplama yükünü tamamlamak için gereken toplam süreyi temsil etmektedir. Uç cihaz aygıtının işlem kapasitesindeki değişiklikler nedeniyle, bölme noktasındaki bellek tüketimi ve enerji tüketimi sırasıyla M_z ve E_z olarak tanımlanmaktadır. Bu, veri iletimi ve işlenmesi sırasındaki toplam enerji harcamasını temsil etmektedir. Optimizasyonda bellek ve enerji tüketiminin önemini belirleyen ağırlık katsayılarına sırasıyla w_1 ve w_2 olarak ifade edilmektedir. Bu katsayılar, optimizasyon süreci sırasında bellek ve enerji tüketimi arasında bir denge kurmak için kullanılmaktadır. Bu nedenle, MH algoritmaları potansiyel bölme noktalarını aday çözümler olarak değerlendirmekte ve amaç fonksiyonunu en aza indiren optimum s_{Opt} değerini seçmektedir. MH algoritmasının amacı, uç-bulut bölünmesinin toplam çıkarım süresini en aza indirecek şekilde gerçekleştirilmesini sağlamaktır. Çıkarım süresi, bant genişliği ve hesaplama yükü gibi değişkenlere dayanarak denklem 10'da gösterildiği gibi hesaplanmaktadır.

$$IT(s_{Opt}) = IT^{f}_{1,s_{Opt}} + \frac{Dc_p(s_{Opt})}{r} + IT^{b}_{s_{Opt}+1,L} \tag{10}$$

$IT^{f}_{1,s_{Opt}}$, uç cihazda s_{Opt} noktasına kadar olan işlem süresini temsil etmektedir. Bu süre, sınırlı işlem gücüne sahip uç aygıt tarafından belirli bir katmana kadar gerçekleştirilen hesaplamaların tamamlanması için gereken süreyi temsil etmektedir. $Dc_p(s_{Opt})$, s_{Opt} katmanındaki sıkıştırılmış veri boyutunu göstermekte ve bu değer, uç cihazdan buluta aktarılacak verinin boyutunu temsil etmektedir. Mevcut ağ bant genişliği, veri aktarım hızını etkileyen önemli bir parametre olan 'r' ile

tanımlanmaktadır. Öte yandan, buluttaki kalan katmanlar için işlem süresi $IT^{b}_{s_{Opt}+1,L}$ ile gösterilmekte ve bu süre, artık bulutta gerçekleştirilen uç cihazda tamamlanamayan hesaplamaların gerçekleştirilmesi için gereken süreyi temsil etmektedir. Burada dikkat edilmesi gereken önemli bir nokta ise metasezgisel algoritmanın s_{Opt} değerini optimize ederek toplam çıkarım süresini en aza indirmesi olmaktadır. Bu optimizasyon, hem uç cihaz hem de bulut kaynaklarının verimli kullanımını sağlamayı amaçlamakta ve MH algoritmaları kullanılarak uygulanabilmektedir.

3.1.1.2. MH Algoritmaları

Bu bölümde önerilen yöntemde kullanılan metasezgisel (MH) algoritmalar ele alınmaktadır. MH algoritmaları, esneklikleri, uyarlanabilirlikleri ve geleneksel matematiksel ve sezgisel tabanlı yaklaşımlara kıyasla yerel optimumlardan kaçma kabiliyetleri nedeniyle optimizasyon problemlerinde giderek daha fazla benimsenmektedir. Kapsamlı alan bilgisi ve önceden tanımlanmış kısıtlamalar gerektiren deterministik matematiksel yöntemlerin aksine, MH algoritmaları açık bir matematiksel modele ihtiyaç duymadan sağlam ve ölçeklenebilir çözümler sağlamaktadır. Ayrıca, genellikle problem özelindeki varsayımlara dayanan sezgisel yöntemlerle karşılaştırıldığında, MH algoritmaları daha genelleştirilmiş bir yaklaşım sunmaktadır. Bu durum, MH algoritmalarını dinamik ve karmaşık optimizasyon problemlerini çözmek için uygun hale getirmektedir. Stokastik yapıları, keşif ve sömürü yeteneklerini dengeleyerek çözüm alanının verimli bir şekilde keşfedilmesine olanak tanımaktadır. Bu, gerçek zamanlı çıkarım ve kaynak tahsisi görevlerinde üstün performansa yol açmaktadır. Önerilen yöntem için dört farklı MH algoritması Gri Kurt Optimizasyonu (GWO) (Nadimi-Shahraki vd., 2024), Parçacık Sürüsü Optimizasyonu (PSO) (Shami vd., 2022), Yapay Arı Kolonisi (ABC) (Kaya vd., 2022) ve Yapay Tavşan Optimizasyonu (ARO) (Anka vd., 2024) seçilmiştir. Gri kurtların sosyal hiyerarşisinden ve avlanma davranışlarından ilham alan GWO, güçlü bir yerel arama mekanizmasını korurken etkili küresel arama yetenekleri de sunmaktadır. Bir arama alanındaki parçacıkların toplu hareketine dayanan PSO, hızlı yakınsama sağlamakta ve özellikle sürekli optimizasyon problemleri için etkili olmaktadır. Arı kolonilerinin yiyecek arama davranışını taklit eden ABC, çözüm

çeşitliliğini korurken erken yakınsamayı etkili bir şekilde önlemektedir. Tavşan hareketlerinden ilham alan ARO ise yüksek boyutlu arama alanlarında keşfi iyileştiren dinamik adaptasyon mekanizmaları sunmaktadır. Bu algoritmalar, her biri çıkarım süresi azaltma, bellek verimliliği ve enerji tüketimini optimize etmeye benzersiz bir katkı sağladığı ve çeşitli ancak tamamlayıcı bir optimizasyon teknikleri seti sunduğu için seçilmektedir. Seçilen algoritmalar, hesaplama verimliliği ve optimizasyon doğruluğu arasında bir denge kurarak, bunları dinamik bölme noktası belirleme problemi için oldukça uygun hale getirmektedir. Bu stratejik seçim, önerilen yöntemin uyarlanabilir, ölçeklenebilir ve verimli kalmasını sağlayarak, CI çerçeveleri içinde DNN bölümleme performansının iyileştirilmesine katkıda bulunmaktadır.

Seçilen MH algoritmaları, bölme noktasını belirlemek ve denklem 9'daki amaç fonksiyonunu en aza indirmeyi hedeflemek için kullanılmaktadır. Başlangıçta, GWO, optimizasyon gerçekleştirmek için gri kurtların avlanma davranışını ve alfa (α), beta (β), delta (δ) ve omega (ω) kurtlarını içeren sosyal hiyerarşilerini simüle etmektedir. Alfa kurtlar arama sürecini yönetirken, beta ve delta kurtlar keşif sürecini yönlendirmektedir. GWO algoritmasının hedef noktaya olan mesafe hesaplaması denklem 11'de gösterilmektedir.

$$D = C.X_{leader} - X_t \tag{11}$$

Bu bağlamda, X_{leader} en iyi çözümü belirten alfa kurdun pozisyonunu temsil ederken, X_t mevcut çözümün pozisyonunu belirtmektedir. D, arama alanındaki çözüm ile en iyi çözüm arasındaki mesafeyi temsil etmekte ve bu mesafe sabit C ile ölçeklenmektedir. Sabit C, $C = 2.r_2$ olarak tanımlanmakta ve r_2 değeri [0, 1] aralığında rastgele seçilmektedir. Böylece, kurtların her yinelemede farklı yönlerde yeni çözüm noktalarını keşfetmesine olanak tanımaktadır. Algoritmadaki her kurt, denklem 12'de gösterildiği gibi, en iyi üç kurdun pozisyonlarına göre pozisyonunu yinelemeli olarak güncellemektedir.

$$X_{new} = \frac{X_1 + X_2 + X_3}{3} \tag{12}$$

Burada, X_1, X_2 , ve X_3 parametreleri kurtların pozisyonlarına göre oluşturulan güncellenmiş pozisyon vektörlerini ifade etmektedir. Güncellenmiş pozisyon bu üç

önde gelen çözümün ortalaması alınarak belirlenmektedir. Her kurdun bireysel pozisyonu denklem 13'de gösterilmektedir.

$$\begin{aligned} X_1 &= X_\alpha - A_1.D_\alpha \ , \\ X_2 &= X_\beta - A_2.D_\beta \ , \\ X_3 &= X_\delta - A_3.D_\delta \end{aligned} \tag{13}$$

Burada, A parametresi, $A = 2a.r_1 - a$ formülü kullanılarak elde edilmekte ve a, iterasyon ilerledikçe 2'den 0'a doğrusal olarak azalmaktadır. r_1 ise [0,1] aralığında rastgele seçilen bir değer olmaktadır. Katsayının zamanla kademeli olarak azaltılması, algoritmanın erken aşamalarda daha geniş bir arama yapmasını ve sonraki aşamalarda çözümleri daha küçük adımlarla ince ayarlamasını sağlamaktadır. PSO, kuşların ve balıkların sürü davranışını modellerken, her parçacığın (aday çözüm) hızını ve konumunu kendi en iyi deneyimine ve küresel olarak en uygun çözüme göre ayarlamaktadır. PSO kullanılarak en uygun çözüme ulaşmak için bir parçacığın hızını güncelleme denklemi, denklem 14'te sunulmaktadır.

$$V_i^{t+1} = w.V_i^t + c_1.r_1.\left(P_i^{best} - X_i^t\right) + c_2.r_2.\left(G_g^{best} - X_i^t\right) \tag{14}$$

Her parçacık, hız vektörü V_i ve en iyi bilinen çözümleri kullanarak yeni bir konum olan X_i'ye hareket etmektedir. V_i^{t+1} denklemi, önceki hız V_i^t, parçacığın kişisel en iyi konumu P_i^{best} ve genel en iyi konumu G_g^{best} temel alınarak hesaplanmaktadır. w parametresi, hareketin yönünü belirleyen atalet ağırlığını temsil ederken, $'c_1'$ ve $'c_2'$ katsayıları sırasıyla kişisel en iyi ve genel en iyi çözümlere doğru hareket etme eğilimini kontrol etmektedir. $'r_1'$ ve $'r_2'$, algoritma içinde keşif ve sömürüyü dengelemede rol oynayan, [0,1] aralığında rastgele seçilen değerleri ifade etmektedir. Parçacığın yeni konumu, hesaplanan hızın mevcut konumuna eklenmesiyle belirlenmektedir. Bu işlem, denklem 15'te gösterilmektedir.

$$X_i^{t+1} = X_t + V_i^{t+1} \tag{15}$$

Yeni konum X_i^{t+1}, güncellenen hız V_i^{t+1}'in geçerli konuma eklenmesiyle elde edilmektedir. ABC algoritmasında, aday bölünme noktaları nektar kaynakları olarak ele alınmakta ve algoritma en uygun bölünme noktasını bulmayı amaçlamaktadır. Algoritma, işçi, gözlemci ve keşifçi arılar olmak üzere üç ana bileşene dayanmaktadır. İşçi arılar geçerli çözümleri araştırırken gözlemci arılar en iyi

çözümleri seçmekte ve keşifçi arılar ise yeni çözümler üretmektedir. ABC algoritması içindeki güncelleme süreci denklem 16'da sunulmaktadır.

$$X_i^{new} = X_i + \phi.(X_i - X_j) \tag{16}$$

Burada, mevcut çözüm X_i ile rastgele seçilmiş komşu çözüm X_j arasındaki fark değerlendirilerek yeni bir çözüm üretilmektedir. Çözüm çeşitliliğini artırmak için katsayı ϕ ise $[-1,1]$ aralığında rastgele seçilmektedir. Uygunluk fonksiyonunun hesaplanması denklem 17'de sunulmaktadır.

$$F_i = \frac{1}{1+f(X_i)} \tag{17}$$

Burada, $f(X_i)$ karşılık gelen çözümün hata veya uygunluk değerini temsil etmektedir. Daha düşük hata değerlerine sahip çözümler, koloninin arıları tarafından daha çok tercih edilmektedir. Böylece, ABC algoritmasında, işçi arılar mevcut çözümleri iyileştirirken, gözlemci arılar en iyi çözümleri seçerek keşif sürecine katkıda bulunmaktadır. Keşif arıları ise zayıf çözümleri terk ederek aramayı yeni bölgelere yönlendirmekte ve böylece yerel minimumlardan kaçınmaya yardımcı olmaktadır. ARO algoritması, optimum bölünme noktasını belirlemek için tavşanların tarama ve av yakalama stratejilerini taklit etmektedir. Bu bağlamda, her tavşan bir aday bölünme noktasını temsil etmekte ve en iyi çözüm yolunu keşfetmek için dinamik bir hareket modeli kullanmaktadır. ARO algoritmasının hareket güncelleme mekanizması denklem 18'de sunulmaktadır.

$$X_i^{new} = X_i + \delta.F(t) \tag{18}$$

Burada, $F(t)$ zamana bağlı bir faktör olmakta ve daha büyük bir δ algoritmanın daha geniş bir arama alanını keşfetmesini sağlamaktadır. Daha küçük bir δ ise en iyi bilinen çözümlere daha fazla odaklanmayı sağlamakta ve böylece optimizasyon sürecini yoğunlaştırmaktadır. Keşif fonksiyonu denklem 19'da sunulmaktadır.

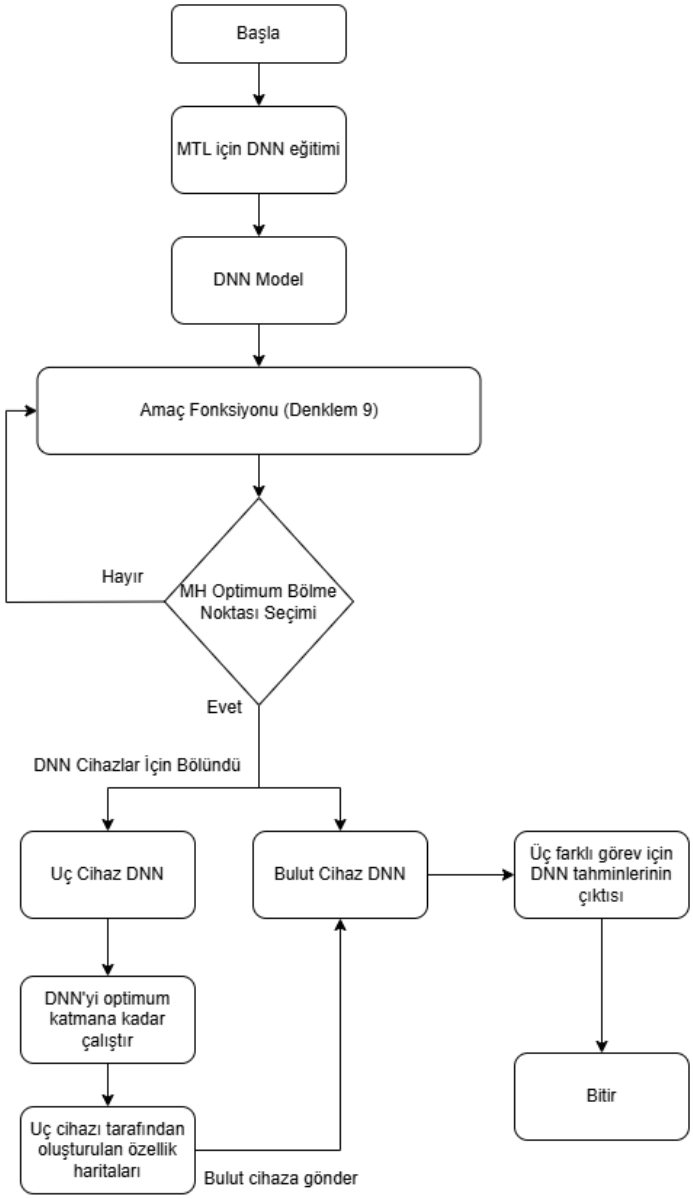
$$F(t) = \left(1 - \frac{t}{T}\right).(X_{best} - X_i) + r.d \tag{19}$$

Bu bağlamda, X_{best} en iyi bulunan çözüm pozisyonunu temsil ederken, r keşif sürecine çeşitlilik getiren rastgele bir sayıyı ifade etmektedir. Katsayı d ise arama yoğunluğunu kontrol ederek algoritmanın hem kapsamlı keşif hem de yakınsama aşamasında ince ayar yapmasına olanak tanımaktadır. Bu mekanizma keşif ve

sömürü arasındaki dengeyi optimize etmeye yardımcı olmaktadır. Sonuç olarak, uygulanan MH algoritmaları en uygun bölme noktalarını belirlemeyi, çıkarım süresini IT_z, enerji tüketimini E_z ve bellek kullanımını M_z iyileştirirken DNN'nin verimli bir şekilde bölümlenmesini sağlamayı amaçlamaktadır.

3.1.2. Önerilen DSPCI-MH Yöntemin Akış Şeması ve Pseudocode

Bu bölümde, önerilen yöntemin çalışma mekanizmasını özetleyen akış şeması ve detaylı sözde kod sırasıyla Şekil 11'de ve Algoritma 2'de sunulmaktadır.



Şekil 11. Önerilen DSPCI-MH yöntemin akış şeması.

Algoritma 2. Önerilen DSPCI-MH Pseudo code olarak gösterimi.

```
# Modeli nicemle
function Quantize_Model(M):
    Q_M = Quantize(M)  # Modele nicemleme uygula
    return Q_M

# Doğal darboğazları bul
function Get_Natural_Bottlenecks(Q_M, input_size):
    Natural_Bottlenecks = []
    for L_i in Q_M.Layers[:-1]:  # Son katman hariç tüm katmanlar üzerinde döngü
        if L_i.output_shape is not None:
            Output_Size = Calculate_Output_Size(L_i.output_shape)  # Çıktı boyutunu hesapla
    return Natural_Bottlenecks

# Çıkarım süresini hesapla
function Get_Inference_Time(Model, Batch_Size, Input_Shape, Repetitions=10):
    Warmup(Model, Batch_Size, Input_Shape, Repetitions)  # Isınma iterasyonları
    Start_Time = Timer()
    for i in range(0, Repetitions):
        Model(Predict(Input_Shape))
    End_Time = Timer()
    Inference_Time = (End_Time - Start_Time) / Repetitions
    return Inference_Time * 100  # Milisaniye cinsinden döndür

# Belirli bir katmandan itibaren kuyruk modelini çıkar
function Get_Tail_Model(M, Start_Layer_Index):
    Tail_Model = Subset_Model(M, Start_Layer_Index, End_Layer=M.End)
    return Tail_Model

# Her bölme noktası için çıkarım süreleri ile bir batch tablosu oluştur
function Create_Batch_Table(M, Config):
    Q_M = Quantize_Model(M)
    Natural_Bottlenecks = Get_Natural_Bottlenecks(Q_M, Config.input_size)
    Batch_Table = {}
    for Batch_Size in Config.batch_sizes:
        Batch_Table[Batch_Size] = {}

function Compute_Memory_Usage(Split_Point): # Bellek kullanımını hesapla
    Memory_Edge = Calculate_Edge_Memory(Split_Point)  # Kenar belleğini hesapla
    Memory_Cloud = Calculate_Cloud_Memory(Split_Point)  # Bulut belleğini hesapla
    return Memory_Edge, Memory_Cloud

function Compute_Energy(Inference_Time, Power_Consumption): # Enerji tüketimini hesapla
    return Inference_Time * Power_Consumption  # Basit enerji modeli

# Amaç fonksiyonu
function Evaluate_Split(Split_Point, Model, Bandwidth, Batch_Size, Power_Consumption):
    Tail_Model = Get_Tail_Model(Model, Split_Point)
    Inference_Time = Get_Inference_Time(Tail_Model, Batch_Size, Input_Shape)
    Memory_Edge, Memory_Cloud = Compute_Memory_Usage(Split_Point)
    Energy_Cost = Compute_Energy(Inference_Time, Power_Consumption)
    Fitness = (w1 * Inference_Time) + (w2 * Memory_Edge) + (w3 * Energy_Cost)  # Uygunluk hesapla
    return Fitness

# Metaheuristik optimizasyon
function Optimize_Split_Point(Model, Metaheuristic_Algorithm, Bandwidth, Batch_Size, Power_Consumption):
    Population = Initialize_Population(Model)  # Aday bölme noktalarını oluştur
    Best_Split = None
    Best_Fitness = ∞

    for iteration in range(0, MAX_ITERATIONS):
        for Split_Point in Population:
            Fitness = Evaluate_Split(Split_Point, Model, Bandwidth, Batch_Size, Power_Consumption)
            if Fitness < Best_Fitness:
                Best_Fitness = Fitness
                Best_Split = Split_Point
        Population = Update_Population(Population, Best_Split, Metaheuristic_Algorithm)

    return Best_Split
```

3.2. ÖNERİLEN DSPCI-MH YÖNTEMİ DENEYSEL SONUÇLAR

Bu kısımda, önerilen DSPCI-MH metodolojisinin daha çeşitli veri kümeleri üzerinde uygulanmasına odaklanılarak iki farklı veri kümesiyle elde edilen sonuçlar sunulmaktadır. Önerilen yöntemin deneysel sonuçları için kullanılan veri setleriyle ilgili açıklamalar kısım 2.2.'de belirtilmektedir. Ayrıca, bölünme noktasını belirlemek için kullanılan dört farklı MH algoritması, bilinen literatürde bulunan diğer yöntemlerle karşılaştırılmaktadır. Bu yöntemler, en verimli performansa göre seçilen ve literatür taraması bölümünde belirtildiği gibi farklı matematiksel ve sezgisel yaklaşımlara dayanan dört farklı yöntemle karşılaştırılmaktadır.

3.2.1. DSPCI-MH için Kurulum ve Yapılandırma

Deneyler, daha önce bahsedilen Tablo 4'te ayrıntılı olarak açıklanan yazılım ve donanım kurulumu kullanılarak yürütülmektedir. Bu yapılandırmalar, hem uç hem de bulut bilişim bileşenleri için donanım özelliklerini ve kullanılan Python ve TensorFlow sürümlerini içermektedir. Uç cihaz, daha düşük kapasiteli bir CPU'ya sahip bir dizüstü bilgisayardır, bulut bileşeni ise uç cihazın CPU'suna kıyasla önemli ölçüde daha fazla hesaplama gücü sunan yüksek performanslı GPU'larla donatılmış uzak bir sunucu tarafından yönetilmektedir. Ek olarak, önerilen yöntemin analizi, değerlendirilmesi ve karşılaştırılması için kullanılan parametreler Tablo 11'de tanımlanmaktadır.

Tablo 11. Analiz ve değerlendirmede kullanılan parametreler.

Parametre	Açıklama
Çıkarım Süresi	Çıkarım süresi, bir model veya algoritmanın bir girdiyi işlemek ve bir çıktı üretmek için harcadığı süre olarak tanımlanmaktadır. Gerçek zamanlı uygulamalar için düşük çıkarım süresi kritik öneme sahip olmaktadır.
Enerji Tüketimi	Enerji tüketimi, bir algoritmanın yürütülmesi sırasında harcanan toplam enerji miktarını ifade etmektedir. Enerji tüketimi, özellikle enerji verimliliğinin önemli olduğu pille çalışan cihazlar ve dağıtılmış sistemler için kritik bir parametre olmaktadır. Daha düşük enerji tüketimi, sistemin daha uzun çalışma ömrüne ve sürdürülebilirliğine katkıda bulunmaktadır.
Bellek Kullanımı	Bellek kullanımı, bir algoritmanın veya modelin yürütülmesi sırasında ihtiyaç duyduğu bellek miktarını ifade etmektedir. Sınırlı kaynaklara sahip sistemlerde, düşük bellek tüketimi, verimli algoritma performansı sağlamak için önemli olmaktadır. Yüksek bellek kullanımı darboğazlara veya sistem arızalarına yol açabilmektedir.
Yürütme Süresi	Yürütme süresi, bir algoritmanın başlatılmasından tamamlanmasına kadar geçen toplam süre olarak tanımlanmaktadır. Bu ölçüm, çıkarım süresini, ön işlemeyi ve diğer hesaplama süreçlerini içermektedir. Büyük veri kümeleriyle çalışan sistemlerde yürütme süresinin optimize edilmesi performans ve verimlilik açısından kritik öneme sahip olmaktadır.

Uygulanan MH algoritmaları için, DNN mimarisine dayalı toplam iterasyon sayısı 20, popülasyon büyüklüğü ise 5 olarak belirlenmiştir. Algoritmaların kendine özgü hiperparametrelerinde ise GWO algoritmasında, keşif-sömürü dengesini düzenleyen 'a' parametresine 2'den 0'a doğrusal olarak azalan bir değer atanmıştır. PSO algoritmasında, 'w' değeri 0,9'dan 0,4'e doğrusal olarak azalan bir değer olarak tanımlanmıştır. Ayrıca, 'c1' ve 'c2' parametrelerinin her ikisi de 2,05 olarak olarak atanmış ve hız sınırları problem uzayına bağlı olarak otomatik şekilde belirlenmiştir. ABC algoritmasında, keşif arısı limiti (popülasyon boyutu * problem boyutu) / 2 olarak atanmıştır. Son olarak, ARO algoritmasında, hızı azaltarak zamanla arama alanını daraltmak için 'δ' değeri 0,04, *'F(t)'* parametresi 0,02 ve yavaşlatma faktörü 0,001 olarak ayarlanmıştır. Belirlenen hiperparametreler, her bir algoritmanın optimizasyon sürecini en iyi şekilde yönetmesini sağlamak amacıyla belirlenmiştir.

3.2.2. MH ile Dinamik Bölünme Noktası Tespiti Sonuçları ve Analizi

Bu bölüm, dinamik bölme noktası seçimi algoritmalarının üç farklı senaryoda karşılaştırmalı performans değerlendirmesini sunmaktadır. Değerlendirme, çeşitli batch boyutları ve bant genişliği değerleri için çıkarım süresi, enerji tüketimi, bellek kullanımı ve yürütme süresi ölçütleri üzerinden gerçekleştirilmiştir. Karşılaştırma için Random (Alvar & Bajic, 2019), Neurosurgeon (Kang vd., 2017), Dinamik Bölme Noktası Hesaplama (DSPC) (Bakhtiarnia vd., 2023) ve Uç Bilişim için Derin Sıçrayıcı Sinir Ağları (EC-SNN) (Yu vd., 2024) seçilmiş olup, bu yöntemler literatürdeki önemleri ve DNN bölümlenmesine yönelik farklı yaklaşımları nedeniyle tercih edilmiştir. Random yöntemi, herhangi bir hesaplama veya parametre gerektirmeden DNN'yi önceden belirlenmiş bir katmanda bölen basit bir yaklaşım olmaktadır. Neurosurgeon, DNN hesaplamalarını uç cihaz ve bulut arasında bölmek için matematiksel bir optimizasyon tekniği kullanarak gecikme süresini ve enerji tüketimini daha verimli hale getiren sistematik bir yöntem sunmaktadır. DSPC yöntemi, hesaplama iş yükü ve ağ koşullarına bağlı olarak bölme noktalarını dinamik şekilde seçebilen daha uyarlanabilir bir mekanizma sağlamaktadır. EC-SNN ise, özellikle gerçek zamanlı ve düşük güç tüketimli AI uygulamaları için enerji verimli bölme hesaplamalarına odaklanan daha yeni bir yaklaşım olmaktadır. Bu üç yöntem, önerilen MH tabanlı yöntemle kapsamlı bir karşılaştırma yapabilmek amacıyla

seçilmiştir. Neurosurgeon, erken dönem matematiksel optimizasyon teknikleri için bir referans noktası niteliğinde olup, DSPC daha dinamik bir bölme hesaplama adaptasyonu sunmaktadır. EC-SNN ise enerji farkındalığı yüksek çıkarım süreçlerine odaklanan bir perspektif sağlamaktadır. Bu yöntemlerin MH tabanlı yaklaşımlarla karşılaştırılması, geleneksel optimizasyon, sezgisel uyarlama ve biyolojik ilhamlı stokastik optimizasyon arasındaki farkları ortaya koyarak MH algoritmalarının dinamik ortamlardaki etkinliğini ve uyarlanabilirliğini göstermektedir. Bu bağlamda, Random, Neurosurgeon, DSPC, EC-SNN, GWO, PSO, ABC ve ARO algoritmalarının performans analizleri gerçekleştirilmiştir. Senaryo 1'de, hızlı veri aktarımını simüle etmek ve ağ tarafında darboğaz oluşumunu önlemek amacıyla batch boyutu 7 ve bant genişliği 15 olarak belirlenmiştir. Senaryo 2, sınırlı bir ağ altyapısını modellemek amacıyla batch boyutunun 7 olarak korunmasına karşın bant genişliği 3'e düşürülerek veri iletim hızının azalmasına neden olacak şekilde tasarlanmıştır. Senaryo 3'te ise daha büyük veri hacimleri oluşturularak veri iletiminde gecikmelerin yaşanması sağlanmış ve batch boyutu 32, bant genişliği ise 3 olarak belirlenmiştir. On bağımsız çalıştırmanın ortalaması alınarak elde edilen deneysel sonuçlar Tablo 12'de özetlenmektedir.

Tablo 12. Deneysel Sonuçlara Genel Bakış.

		Random	**Neuro surgeon**	**DSPC**	**EC-SNN**	**DSP-GWO**	**DSP-PSO**	**DSP-ABC**	**DSP-ARO**
Senaryo 1	IT (ms)	238.278	213.583	195.543	67.8	0.738	**0.612**	0.656	0.763
	Energy (μJ)	119139	106791.5	97771.5	33900	369.27	**306**	328	381
	Memory	4.01 MB	2.31 MB	4.01 MB	138.1 MB	9.168 KB	18.724 KB	**3.032 KB**	4.216 KB
	Speed (sn)	2970.8 ms	530	261.54	240.211	13.186	12.902	13.398	**12.841**
Senaryo 2	IT (ms)	277.32	254.67	419.26	50.60	**0.593**	0.633	0.881	0.798
	Energy (μJ)	138660	128335	209630	25300	**296.9**	316	440	399
	Memory	4.01 MB	2.39 MB	4.21 MB	138.5 MB	**2.969 KB**	31.54 KB	23.748 KB	56.89 KB
	Speed (sn)	3020.4 ms	540.16	492.23	238.19	45.549	13.056	**12.693**	12.844
Senaryo 3	IT (ms)	1311	Out of memory			3.007	**2.063**	2.501	2.654
	Energy (μJ)	655727				1503	**1030**	1250	1320
	Memory	19.26 MB				7.56 KB	36.636 KB	8.556 KB	**3.48 KB**

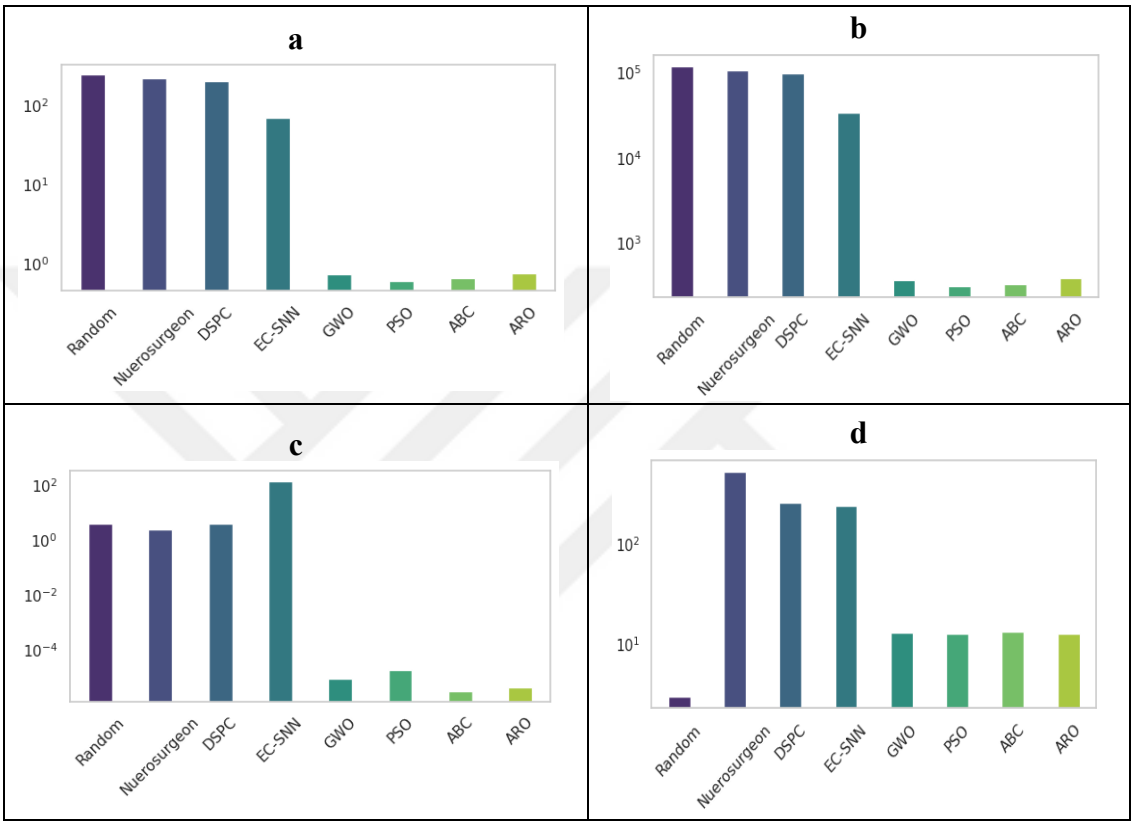
Speed (sn)	5522.4 ms	28,594	33.541	21.646	**18.954**	

En iyi değerler vurgulu ve kalındır.

3.2.2.1. Senaryo 1

Senaryo 1 kapsamında gerçekleştirilen analizlerden elde edilen bulgular, DSP-GWO, DSP-PSO, DSP-ABC ve DSP-ARO yaklaşımlarının, batch boyutu 7 ve bant genişliği 15 olan durumda sırasıyla 0.738, 0.612, 0.656 ve 0.763 ms olmak üzere önemli ölçüde daha kısa çıkarım süreleri sağladığını göstermektedir. Bu sonuçlar, MH algoritmalarının diğer yöntemlere kıyasla belirgin şekilde daha verimli bir performans sunduğunu ortaya koymaktadır. Bununla birlikte, EC-SNN yöntemi 67.8 ms ile oldukça yavaş bir çıkarım süresi sergilerken, Random, Neurosurgeon ve DSPC yöntemleri sırasıyla 238.278 ms, 213.582 ms ve 195.543 ms ile daha da uzun sürelerde çıkarım gerçekleştirmiştir. Enerji tüketimi açısından değerlendirildiğinde, PSO 306 µJ ile en düşük enerji kullanımına sahipken bunu sırasıyla 369.27 µJ ve 328 µJ ile GWO ve ABC takip etmektedir. Bu durum, MH algoritmalarının enerji verimliliği açısından diğer yöntemlere kıyasla önemli avantajlar sunduğunu göstermektedir. Bölme noktası seçiminde herhangi bir analiz yapmayan Random yöntemi, 119139 µJ ile en yüksek enerji tüketimini gerçekleştirmektedir. Benzer şekilde, Neurosurgeon ve DSPC yöntemleri de sırasıyla 106791.5 µJ ve 97771.5 µJ ile nispeten yüksek enerji tüketimi sergilemektedir. Bellek kullanımı bağlamında değerlendirildiğinde, EC-SNN 138.1 MB ile en yüksek bellek tüketimine sahip olurken GWO, PSO, ABC ve ARO yöntemleri 0.003 MB ile 0.018 MB arasında değişen oldukça düşük bellek tüketimi sergilemektedir. Random ve DSPC yöntemleri ise 4.01 MB ile orta düzeyde bellek kullanımına sahip olmaktadır. Bu bulgular, algoritma tasarımı ve optimizasyon stratejilerinin bellek tüketimi üzerinde önemli bir etkisi olduğunu vurgulamaktadır. Özellikle ABC'nin düşük bellek kullanımı, sınırlı kaynaklara sahip sistemler için tercih edilen bir seçenek olmasını sağlamaktadır. Yürütme süresi açısından değerlendirildiğinde ise herhangi bir ön işleme ve hesaplama gerektirmeyen Random yöntemi 2970.8 ms ile en hızlı sonucu elde etmektedir. ARO ve PSO ise sırasıyla 12.841 ve 12.902 saniye ile benzer yürütme süreleri göstermektedir. En uzun yürütme süresi ise 530 saniye ile Neurosurgeon yöntemine ait olmaktadır. Bu sonuçlar, farklı ölçütler açısından

performansın uygulama gereksinimlerine bağlı olarak önemli ölçüde değişebileceğini göstermektedir. Senaryo 1'e ait karşılaştırmalı grafik Şekil 12'de sunulmuş olup, bu bulgular, algoritmaların performanslarının farklı ölçütlere göre uygulama bağlamına ve sistem kısıtlarına bağlı olarak değişebileceğini ortaya koymaktadır.



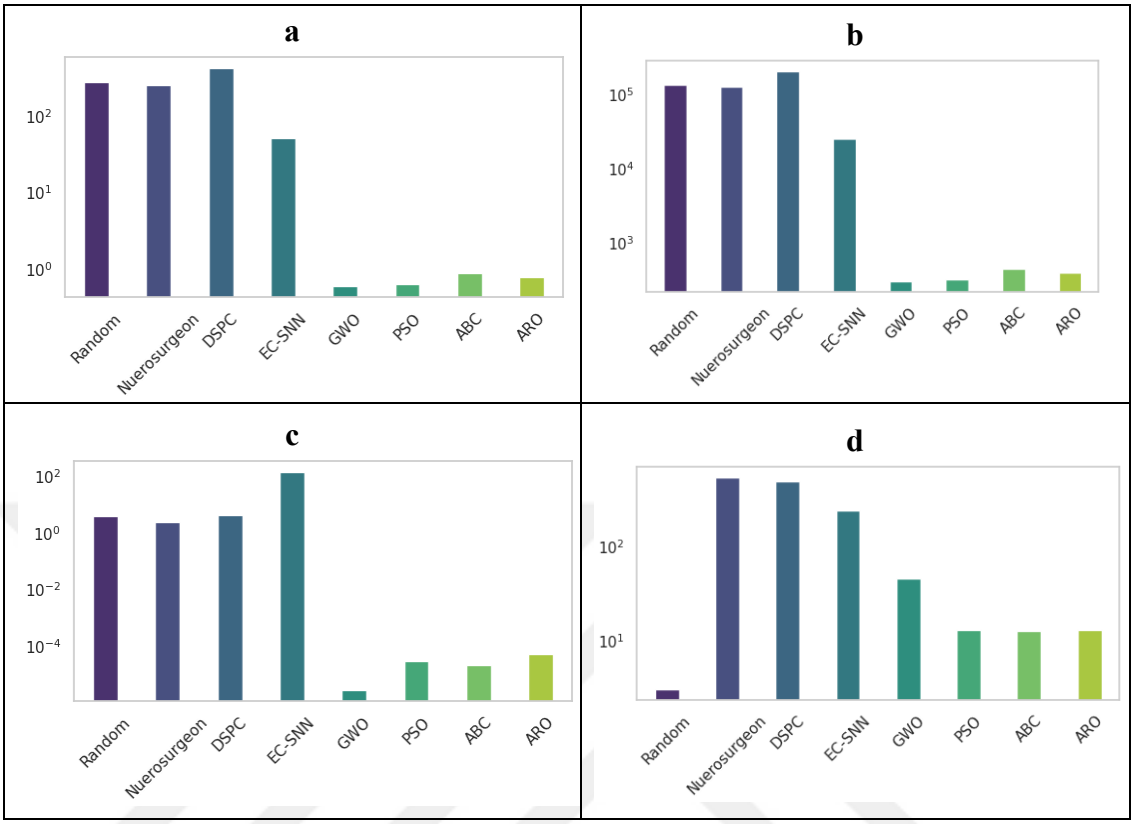
Şekil 12. Senaryo 1 için Performans Ölçütleri. a) Çıkarım Süresi (ms), b) Enerji Tüketimi (µJ), c) Bellek Kullanımı (MB), d) Yürütme Süresi (sn).

Bu senaryoda, Random yöntemi yürütme hızı açısından daha verimli olmakla birlikte, optimal bölme noktası belirlendikten sonra enerji tüketimi, bellek kullanımı ve DNN çıkarım süresi açısından önemli ölçüde dezavantajlı hale gelmektedir. Öte yandan, matematiksel ve sezgisel yöntemler belirli iyileştirmeler sağlasa da enerji ve bellek tüketimi açısından hâlâ bazı dezavantajlara sahip olmaktadır. Buna karşın, uygulanan MH algoritmaları diğer yöntemlere kıyasla daha verimli bir performans sunmaktadır. Özellikle, DSP-PSO çıkarım süresi ve enerji tüketimi açısından öne çıkarken, ABC bellek kullanımı bakımından üstün performans sergilemektedir. Ayrıca, DSP-PSO ve DSP-ARO yürütme süresi açısından daha başarılı sonuçlar elde etmektedir. Bu bulgular, birden fazla performans ölçütü arasında denge gerektiren

görevler için MH tabanlı algoritmaların üstünlüğünü daha da desteklemektedir. Özellikle, hem enerji verimliliğinin hem de yürütme hızının kritik olduğu senaryolarda, MH algoritmalarının etkinliği belirgin şekilde ortaya çıkmaktadır.

3.2.2.2. Senaryo 2

Daha önce belirtildiği üzere ikinci senaryoda daha yavaş veri aktarımı sağlamak amacıyla bant genişliği 3'e düşürülmekte ancak batch boyutu aynı tutulmaktadır. Çıkarım süresi açısından değerlendirildiğinde, DSP-GWO, DSP-PSO, DSP-ABC ve DSP-ARO sırasıyla 0.593, 0.633, 0.881 ve 0.798 ms ile en düşük süreleri elde ederek en verimli sonuçları sunmaktadır. Buna karşın, EC-SNN algoritması 50.6 ms, DSPC ise 419.26 ms ile en yüksek çıkarım süresine sahip olmaktadır. Enerji tüketimi bağlamında incelendiğinde, DSP-GWO algoritması 296.9 μJ ile en düşük enerji kullanımını sağlamış, onu sırasıyla 316 μJ ve 399 μJ ile DSP-PSO ve DSP-ARO algoritmaları takip etmektedir. Bu bulgular, bu algoritmaların benzer bir verimlilik sunduğunu göstermektedir. Öte yandan, DSPC algoritması 209630 μJ ile en yüksek enerji tüketimini sergilemektedir. Bant genişliğinin azalması bellek kullanımını da etkilemiş olup, EC-SNN 138.53 MB ile en yüksek bellek tüketimine sahip olmaktadır. DSP-GWO, DSP-PSO, DSP-ABC ve DSP-ARO algoritmaları ise 0.002-0.056 MB arasında değişen düşük bellek tüketimi göstermektedir. Yürütme süresi açısından değerlendirildiğinde ise DSP-ABC algoritması 12.693 saniye ile en hızlı performansı sergilerken Neurosurgeon algoritması 540.16 saniye ile en yavaş yürütme süresine sahip olmaktadır. Bu senaryo, bant genişliğinin azalmasının özellikle enerji tüketimi ve çıkarım süresi üzerinde önemli bir etkiye sahip olduğunu vurgulamaktadır. Bant genişliği kısıtlaması, algoritmaların daha sınırlı kaynaklara uyum sağlamasını gerektirmektedir. Senaryo 2'ye ait yöntemlerin karşılaştırmalı grafiği Şekil 13'te sunulmaktadır.



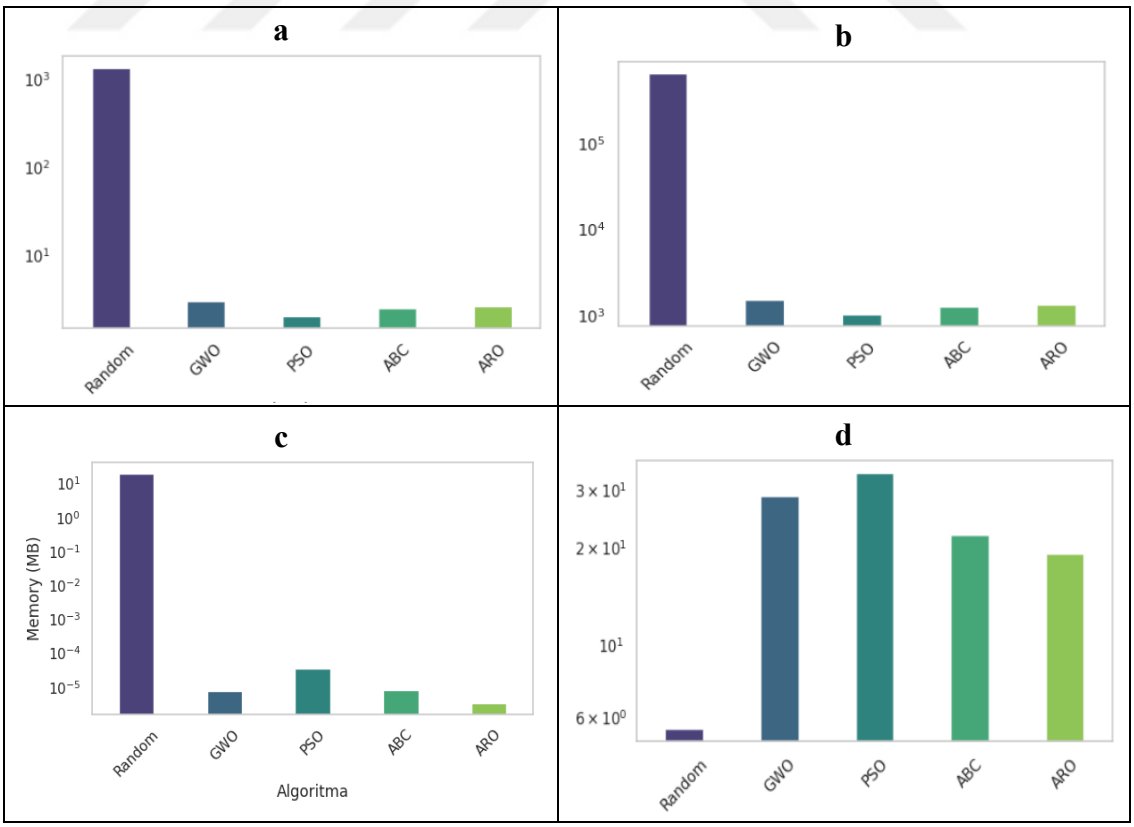
Şekil 13. Senaryo 2 için Performans Ölçütleri. a) Çıkarım Süresi (ms), b) Enerji Tüketimi (µJ), c) Bellek Kullanımı (MB), d) Yürütme Süresi (sn).

İkinci senaryoda, MH algoritmaları diğer yöntemlere kıyasla daha verimli performans göstermektedir. Çıkarım süresi, enerji tüketimi ve bellek kullanımı açısından GWO daha iyi performans göstermektedir. Yürütme hızı açısından ise ABC daha verimli olmaktadır.

3.2.2.3. Senaryo 3

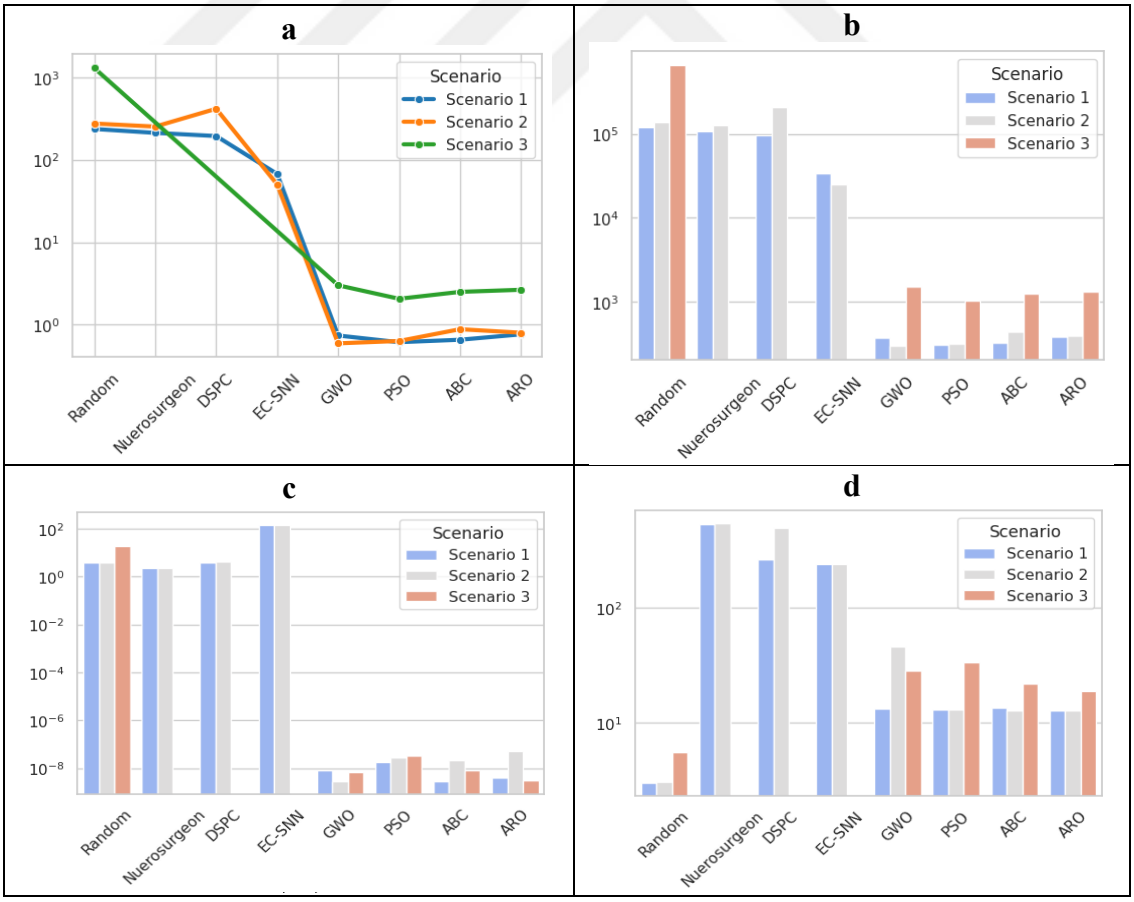
Üçüncü senaryoda, veri iletimindeki gecikmeleri simüle etmek amacıyla batch boyutu 32'ye yükseltilerek bant genişliği 3 olarak sabitlenmiştir. Ayrıca, bu senaryoda, bellek yetersizliği hataları nedeniyle Neurosurgeon, DSPC ve EC-SNN yöntemleri için herhangi bir sonuç elde edilememiştir. Özellikle büyük veri hacimlerinin iletildiği durumlarda ortaya çıkan bu sorun, ilgili yöntemlerin verimliliğini olumsuz etkilemektedir. Çıkarım süresi açısından değerlendirildiğinde, DSP-GWO, DSP-PSO, DSP-ABC ve DSP-ARO sırasıyla 3.007, 2.063, 2.501 ve 2.654 ms ile en düşük süreleri elde etmektedir. Buna karşılık, Random yöntemi 1311.455 ms ile en uzun çıkarım süresine sahip olmaktadır. Enerji tüketimi açısından

incelendiğinde, PSO algoritması 1030 µJ ile en düşük enerji tüketimini göstermektedir. Bunu sırasıyla 1503 µJ ve 1320 µJ ile DSP-GWO ve DSP-ARO algoritmaları takip etmektedir. Öte yandan, Random yöntemi 655727 µJ ile en yüksek enerji tüketimine sahip olmaktadır. Bellek kullanımı bağlamında değerlendirildiğinde, MH algoritmaları arasında en düşük bellek tüketimi 0.003 MB ile DSP-ARO algoritmasına ait olmaktadır. Yürütme süresi açısından ise, DSP-ARO algoritması 18.954 saniye ile en hızlı sonucu sağlarken Random yöntemi 5.5224 saniye ile en uzun yürütme süresine sahip olmaktadır. Bu senaryoda, yığın boyutunun artırılması, çıkarım süresi ve enerji tüketimi üzerinde önemli bir etkiye sahip olmakla beraber geleneksel yöntemlerde bellek kullanımıyla ilgili sorunları da ortaya çıkarmaktadır. Bu nedenle, MH algoritmalarının bu sorunu verimli bir şekilde ele alması kritik öneme sahip olmaktadır. Çıkarım süresi ve enerji tüketimi açısından DSP-PSO daha verimli bir performans sergilerken bellek tüketimi ve yürütme süresi açısından DSP-ARO daha yüksek verimlilik sunmaktadır. Senaryo 3'e ait yöntemlerin karşılaştırmalı grafiği Şekil 14'te sunulmaktadır.



Şekil 14. Senaryo 3 için Performans Ölçütleri. a) Çıkarım Süresi (ms), b) Enerji Tüketimi (µJ), c) Bellek Kullanımı (MB), d) Yürütme Süresi (sn).

Bu doğrultuda, elde edilen sonuçlar algoritmaların performansının farklı senaryolar arasında değişkenlik gösterdiğini ve bölme noktasının belirlenmesi sürecinde en verimli yöntemin seçilmesinin kritik önem taşıdığını ortaya koymaktadır. Şekil 15, senaryoya dayalı karşılaştırmalı bir grafiği sunmaktadır. Özellikle DSP-GWO, DSP-PSO, DSP-ABC ve DSP-ARO olmak üzere MH algoritmaları çıkarım süresi, enerji tüketimi, bellek kullanımı ve yürütme hızı açısından öne çıkmaktadır. EC-SNN, çıkarım süresi ve enerji tüketimi açısından MH algoritmalarına kıyasla daha yavaş olmasına rağmen diğer geleneksel algoritmalara göre daha verimli bir performans sergilemektedir. Buna karşılık, Neurosurgeon ve DSPC algoritmaları daha yüksek enerji tüketimi ve yürütme süreleriyle düşük performans göstermektedir. Bu bulgular, özellikle gelecekteki verimliliğin sağlanabilmesi için bölme noktası seçilirken bu faktörlerin dikkate alınmasının önemini vurgulamaktadır.



Şekil 15. Senaryolara Göre Karşılaştırmalı Sonuçlar. a) Çıkarım Süresi (ms), b) Enerji Tüketimi (µJ), c) Bellek Kullanımı (MB), d) Yürütme Süresi (sn)

3.3. DSPCI-MH DEĞERLENDİRME

3.3.1. Tartışma

Bu bölümde, önerilen DSPCI-MH yöntemi ayrıntılı olarak ele alınmaktadır. Deneysel sonuçlar, dinamik ayrım noktası belirlemenin farklı yığın boyutları ve bant genişliği koşullarında performansı önemli ölçüde etkilediğini açıkça ortaya koymaktadır. Çıkarım süresi, enerji tüketimi, bellek kullanımı ve yürütme süresi açısından değerlendirildiğinde, MH algoritmalar, geleneksel ve sezgisel tabanlı yöntemlere kıyasla tutarlı şekilde daha yüksek verimlilik göstermektedir. Özellikle GWO, PSO, ABC ve ARO algoritmaları, çıkarım süresi verimliliğini geleneksel yaklaşımlara kıyasla %99,69 ila %99,86 oranında artırmaktadır. Bu algoritmalar arasında GWO ve PSO en kısa çıkarım sürelerine ulaşmakta olup, gecikmenin kritik olduğu gerçek zamanlı uygulamalar için özellikle etkili olmaktadır. Buna karşılık, EC-SNN, Neurosurgeon ve DSPC yöntemleri, büyük veri hacimleri ve sınırlı bant genişliği altında zorluk yaşamakta ve bu durum daha uzun çıkarım sürelerine ve aşırı durumlarda bellek taşmalarına yol açmaktadır. Enerji tüketimi açısından değerlendirildiğinde, MH tabanlı yöntemler %99,68 ila %99,85 arasında değişen oranlarda azalma sağlamaktadır. Tüm senaryolarda en düşük enerji tüketimi GWO ve PSO algoritmalarında gözlemlenmiştir; bu da onları batarya ile çalışan ve enerji duyarlı IoT ortamları için uygun hale getirmektedir. Buna karşılık, Random, Neurosurgeon ve DSPC yöntemleri oldukça yüksek enerji tüketimi göstermekte ve bu durum onları pratik edge computing uygulamaları için elverişsiz kılmaktadır. Ayrıca, üçüncü senaryoda büyük yığın boyutlarının söz konusu olduğu durumlarda, geleneksel yöntemler bellek kısıtları nedeniyle başarısız olmakta ve böylece MH dışı yaklaşımların ölçeklenebilirlik sınırlarını göstermektedir. Bellek kullanımı açısından bakıldığında, MH algoritmaları geleneksel yöntemlere kıyasla %98,68 ila %99,98 arasında bellek verimliliği artışı sağlamaktadır. Bu başarı, özellikle bellek sınırlamaları nedeniyle cihaz kapanmalarını önleme yeteneklerinden kaynaklanmaktadır. EC-SNN tüm senaryolarda aşırı bellek tüketimi göstermekte olup, kaynakları sınırlı cihazlar için verimli olmamaktadır. Öte yandan, GWO, PSO, ABC ve ARO algoritmaları düşük bellek tüketimi ile gömülü sistemler ve kenar

bilişim uygulamaları için yüksek potansiyel sunmakta ve bu durum onları, bellek sınırlamalarının belirleyici olduğu gerçek zamanlı yapay zekâ temelli IoT dağıtımları için oldukça faydalı kılmaktadır. Yürütme süresi bakımından, Random yöntemi optimizasyon içermediği için en kısa sürede tamamlamakta ancak diğer metriklerdeki zayıf performansı, bu yöntemi gerçek dünya uygulamaları için işlevsiz hâle getirmektedir. MH algoritmaları arasında ARO ve PSO en kısa yürütme sürelerine ulaşmakta ve hesaplama yükü açısından en verimli çözümler olarak öne çıkmaktadır. Buna karşın, Neurosurgeon ve DSPC özellikle büyük ölçekli senaryolarda oldukça uzun yürütme süreleri gerektirmekte ve bu durum onların dinamik ve gerçek zamanlı uygulamalara uyarlanabilirliğini sınırlandırmaktadır. Genel olarak MH algoritmaları, yürütme süresi verimliliğinde %98,49 ila %99,65 arasında iyileşme sağlamakta ve onları adaptif derin öğrenme çıkarımı açısından üstün bir alternatif hâline getirmektedir. Senaryo bazlı analiz yapıldığında ise MH tabanlı dinamik ayrım noktası belirlemenin güçlü yönleri daha da belirginleşmektedir. Senaryo 1'de (yüksek bant genişliği ve küçük yığın boyutları) MH algoritmaları, çıkarım süresi ve enerji tüketimi açısından olağanüstü verimlilik sergilemektedir. Senaryo 2'de, azaltılmış bant genişliği iletişim yükünü artırsa da MH algoritmaları hâlâ geleneksel yöntemlerin önüne geçmektedir. Senaryo 3'te, artan yığın boyutu ciddi bellek sorunlarına yol açmakta ve geleneksel yöntemler tamamen başarısız olmaktadır. Ancak MH algoritmaları bu senaryoya başarıyla uyum sağlayarak hem verimlilik hem de ölçeklenebilirlik sunmaktadır. Genel olarak elde edilen sonuçlar, MH tabanlı ayrım noktası belirlemenin, geleneksel sabit ya da sezgisel yöntemlere kıyasla dinamik, ölçeklenebilir ve verimli bir alternatif sunduğunu ortaya koymaktadır. Gerçek zamanlı koşullara uyum sağlama, enerji ve bellek kullanımını optimize etme ve çıkarım gecikmelerini en aza indirme kabiliyeti sayesinde MH algoritmaları, kenar-bulut CI çerçeveleri için oldukça uygun olmaktadır. Gelecek çalışmalarda, hibrit MH yaklaşımları, dinamik hiperparametre ayarlamaları ve pekiştirmeli öğrenme destekli MH modelleri gibi yöntemler araştırılarak derin öğrenme uygulamalarında ayrım noktası seçiminin daha da iyileştirilmesi hedeflenmelidir.

3.3.2. Sonuç

Bu çalışma, CI mimarileri içerisinde DNN için ayrım noktalarının dinamik olarak belirlenmesine yönelik yeni bir yaklaşım olan DSPCI-MH çerçevesini tanıtmaktadır. Önerilen yöntem, GWO, PSO, ABC ve ARO gibi MH algoritmaları entegre ederek, E2C ve E2E dağıtık hesaplama ortamlarında performans iyileştirmeyi amaçlamaktadır. Farklı senaryolarda yapılan deneysel sonuçlar, MH tabanlı ayrım noktası belirlemenin, rastgele, matematiksel ve sezgisel tabanlı bölme yöntemleri de dâhil olmak üzere geleneksel yaklaşımların tamamından üstün olduğunu tutarlı bir şekilde ortaya koymuştur. Elde edilen bulgular, önerilen DSPCI-MH yönteminin çıkarım süresini önemli ölçüde azalttığını, enerji tüketimini optimize ettiğini ve bellek verimliliğini artırdığını göstermektedir. Bu özellikler, yöntemi yapay zekâ destekli gerçek zamanlı IoT uygulamaları için etkili bir çözüm hâline getirmektedir. Uygulanan algoritmalar arasında PSO, en yüksek çıkarım verimliliğini gösterirken, ARO ise bellek kullanımını optimize ederek MH algoritmalarının dinamik ağ koşulları ve hesaplama kısıtlarına uyum yeteneğini daha da güçlendirmektedir. Tüm avantajlarına rağmen, DSPCI-MH çerçevesi MH algoritmalarının parametre ayarlarına duyarlılığı sebebiyle performansı ağ koşulları, yığın boyutu değişimleri ve hesaplama sınırlamaları gibi faktörlerden etkilenmektedir. Bu zorlukların ele alınması, önerilen yaklaşımın dayanıklılığını ve genellenebilirliğini daha da artıracaktır. Gelecek çalışmalarda, DSPCI-MH çerçevesinin uyarlanabilirlik, ölçeklenebilirlik ve verimlilik açısından aşağıdaki yönlerinin incelenmesi önerilmektedir:

- MH parametrelerinin manuel ayarlanmasına olan ihtiyacı ortadan kaldırmak için, pekiştirmeli öğrenme (RL) ya da uyarlanabilir optimizasyon teknikleri kullanılarak otomatik hiperparametre ayarlama mekanizmalarının entegrasyonu sağlanabilir.
- DSPCI-MH, transformatör tabanlı yapılar ve federe öğrenme senaryoları dâhil olmak üzere ultra büyük ölçekli derin öğrenme modellerine uyarlanarak genişletilebilir.
- Evrimsel algoritmalar ile derin pekiştirmeli öğrenmeyi birleştiren hibrit MH stratejilerinin entegre edilmesi, çerçevenin değişken ağ koşullarına dinamik biçimde uyum sağlama kapasitesini artırabilir.

DÖRDÜNCÜ BÖLÜM

4. GENEL DEĞERLENDİRME

Bu kısımda, önerilen yöntemlerin tartışması ve limitlerinden bahsedilmektedir.

4.1. TARTIŞMA

Bu kısımda, önerilen DSPCI-MTL ve DSPCI-MH yöntemleri detaylı bir şekilde ele alınarak tartışılmaktadır. IoT cihazlarında DNN uygulanması, kısıtlı donanım kapasitesi, sınırlı bant genişliği, uzun veri aktarım süreleri ve iletim hatalarından kaynaklanan görüntü paketi kaybı olmak üzere önemli zorluklar içermektedir. Bu sorunların üstesinden gelmek amacıyla DSPCI-MTL yöntemi önerilmektedir. Önerilen yöntem, üç farklı DNN görev alanında eğitilerek E2E-E2C cihazlarında dağıtılmaktadır. 1-7 MBps bant genişliği koşulları ve 1 ila 15 arasında değişen batch boyutları altında gerçekleştirilen performans değerlendirmeleri, bireysel ağ katmanlarının işlem verimliliğinin farklı ağ kısıtları altında önemli ölçüde değiştiğini ortaya koymaktadır. Özellikle, batch boyutu üçü aştığında ve bant genişliği sınırlı olduğunda, ‘block3_pool’ katmanı en kısa çıkarım süresini göstermektedir. Buna karşılık, daha yüksek bant genişliklerinde bu katman üstün işlem hızları sağlamaktadır. Benzer şekilde, ‘block4_pool’ ve ‘block5_pool’ katmanları düşük bant genişliği koşullarında daha iyi çıkarım süreleri sunmaktadır. Çalışmanın temel bulgularından biri olan ağın ‘block3_pool’ katmanında bölünmesinin, tüm işlemlerin yalnızca uç cihazda gerçekleştirildiği yapılandırmalara kıyasla toplam işlem süresini %38’e kadar azaltmaktadır. Bununla birlikte, bölme katmanının seçimi, rastgele bölme noktası seçimi, çıkarım süresi, batch boyutu ve ağ koşulları faktörlerine dayalı olarak dinamik şekilde optimize edilmesi gerekmektedir. Bulgular, farklı batch boyutları ve bant genişliği ayarları arasında bölme işlemlerinden elde edilen dinamik kazancı daha ayrıntılı şekilde ortaya koymaktadır. Yapılan analizler, uçtan buluta aktarımın olmadığı senaryolara kıyasla işlem

süresinde %61'e varan iyileşmeler sağlandığını ve 'fc1', 'block3_pool', 'block4_pool' ve 'block5_pool' katmanlarında önemli performans artışları elde edildiğini göstermektedir. Buna ek olarak, önerilen AE yöntemi, iletim sırasında özellik haritası içindeki benzerlikleri temel alarak veri yeniden yapılandırması için tasarlanarak geleneksel yöntemlerden farklılaşmaktadır. AE yöntemi, geleneksel CALTeC, HaLRTC ve SiLRTC yöntemlerinden üstün performans göstererek 32.37 PSNR ve 0.851 SSIM değerlerine ulaşmaktadır. Bu yenilikçi yaklaşım, değerlendirilen üç farklı görev kapsamında kaybolan paketleri etkili bir şekilde yeniden yapılandırarak DNN tahmin doğruluğunu artırmaktadır.

Önerilen DSPCI-MH yönteminde deneysel sonuçlar, dinamik bölme noktası belirlemenin değişen batch boyutları ve bant genişliği koşulları altında performansı önemli ölçüde etkilediğini açıkça göstermektedir. Çıkarım süresi, enerji tüketimi, bellek kullanımı ve yürütme süresi açısından değerlendirildiğinde MH algoritmalarının verimlilik açısından geleneksel ve sezgisel tabanlı yöntemlere kıyasla üstün performans sergilediği görülmektedir. Özellikle GWO, PSO, ABC ve ARO algoritmaları, geleneksel yaklaşımlara kıyasla çıkarım süresi verimliliğini %99.69 ila %99.86 oranında artırmaktadır. Bu algoritmalar arasında, GWO ve PSO en hızlı çıkarım sürelerini sunarak gecikme hassasiyeti yüksek gerçek zamanlı uygulamalar için özellikle etkili hale gelmektedir. Buna karşılık, EC-SNN, Neurosurgeon ve DSPC yöntemleri, büyük veri hacimleri ve sınırlı bant genişliği altında zorlanmakta ve bu durum daha uzun çıkarım sürelerine ve aşırı durumlarda bellek tükenmesine neden olmaktadır. Enerji tüketimi açısından değerlendirildiğinde, MH tabanlı yöntemler %99.68 ila %99.85 oranında azalma sağlamaktadır. GWO ve PSO, tüm senaryolarda en verimli enerji tüketimini göstererek, batarya ile çalışan ve enerji açısından duyarlı IoT ortamları için uygun hale gelmektedir. Öte yandan, Rastgele, Neurosurgeon ve DSPC yöntemleri önemli ölçüde daha yüksek enerji tüketimi göstermekte olmakta ve bu durum uç bilişim uygulamaları için verimi azaltmaktadır. Büyük veri hacmi boyutlarının kullanıldığı üçüncü senaryoya benzer durumlarda geleneksel yöntemlerin bellek kısıtlamaları nedeniyle başarısız olduğu görülmekte ve böylece MH tabanlı olmayan yaklaşımların ölçeklenebilirlik açısından sınırlamaları pekiştirilmektedir. Bellek kullanımı açısından değerlendirildiğinde, MH algoritmaları, geleneksel yöntemlere kıyasla %98.68 ila %99.98 oranında bellek

verimliliği artışı sağlamaktadır. Bu iyileşme büyük ölçüde, bellek kısıtlamaları nedeniyle cihazın devre dışı kalmasını önleme yeteneklerinden kaynaklanmaktadır. EC-SNN, tüm senaryolarda aşırı bellek tüketimi göstererek kaynak açısından kısıtlı cihazlar için verimsiz hale gelmektedir. Buna karşılık, GWO, PSO, ABC ve ARO, daha verimli bellek tüketimleriyle gömülü ve uç bilişim uygulamaları için yüksek potansiyel taşımakta olmakta ve özellikle bellek kısıtlarının belirleyici olduğu gerçek zamanlı yapay zeka destekli IoT dağıtımları için faydalı olmaktadır. Yürütme süresi açısından incelendiğinde ise optimizasyon içermeyen yapısı nedeniyle Rastgele yöntemin en hızlı sonuç verdiği görülmektedir. Ancak, diğer metriklerdeki düşük performansı nedeniyle gerçek dünya uygulamaları için verimli olmamaktadır. MH algoritmaları arasında, ARO ve PSO en kısa yürütme sürelerini sunarak hesaplama yükü açısından en verimli yöntemler olarak öne çıkmaktadır. Buna karşılık, büyük ölçekli senaryolarda Neurosurgeon ve DSPC yöntemleri önemli ölçüde daha uzun yürütme sürelerine ihtiyaç duymakta olmakta ve bu durum onları dinamik gerçek zamanlı uygulamalar için daha az uyarlanabilir hale getirmektedir. MH algoritmalarının genel yürütme süresi verimliliği %98.49 ila %99.65 arasında iyileşme sağlaması adaptif derin öğrenme çıkarımı konusundaki üstünlüklerini ortaya koymaktadır. MH tabanlı dinamik bölme noktası belirleme yönteminin güçlü yönleri senaryolara dayalı analizlerle daha da öne çıkmaktadır. Senaryo 1'de, yüksek bant genişliği ve küçük veri hacmi boyutları kullanıldığında MH algoritmaları hem çıkarım süresi hem de enerji tüketimi açısından olağanüstü verimlilik sergilemektedir. Senaryo 2'de, bant genişliği kısıtlarının artırılması iletişim yükünü artırsa da MH algoritmaları geleneksel yöntemlere kıyasla üstün verimlilik sağlamaktadır. Senaryo 3'te, veri hacminin artması, geleneksel yöntemlerin bellek tükenmesi nedeniyle tamamen başarısız olmasına yol açmaktadır. Ancak MH algoritmaları bu duruma başarılı bir şekilde uyum sağlayarak hem verimlilik hem de ölçeklenebilirlik açısından avantaj sunmaktadır. Genel olarak, elde edilen sonuçlar, MH tabanlı bölme noktası belirleme yönteminin geleneksel statik veya sezgisel tabanlı yöntemlere kıyasla dinamik, ölçeklenebilir ve verimli bir alternatif sunduğunu vurgulamaktadır. Gerçek zamanlı koşullara uyum sağlama, enerji ve bellek kullanımını optimize etme ve çıkarım gecikmelerini en aza indirme yeteneği, MH algoritmalarını uç-bulut CI çerçeveleri için son derece uygun hale getirmektedir.

Gelecekteki çalışmalar, hibrit MH yaklaşımlarını, dinamik hiperparametre ayarlama tekniklerini ve pekiştirmeli öğrenme destekli MH modellerini inceleyerek derin öğrenme uygulamalarında bölme noktası seçim süreçlerini daha da geliştirmeye odaklanması gerekmektedir.

4.2. LİMİTLER

Bu kısım, tez çalışmasında önerilen DSPCI-MTL ve DSPCI-MH yöntemlerine ait sınırlamaları ele almaktadır. DSPCI-MTL için optimal bölme noktasının belirlenmesinde iki temel zorluk ortaya çıkmaktadır. Bunlardan ilki uç cihazın donanım kısıtlamaları sebebiyle özellikle son derece büyük veri hacimleri işlenirken arabellek taşma eşiği oluşmaktadır. Bir diğer zorluk ise DNN'in karmaşık yapısı optimal bölme noktasının belirlenmesi sürecinde çıkarım süresini olumsuz yönde etkileyebilmektedir. Ayrıca, özellik haritası içindeki benzerlik oranına dayalı eksik görüntü paketlerini kurtarma yöntemi doğası gereği bazı sınırlamalara sahip olmaktadır. Özellikle, önerilen yöntem, farklı kontrast seviyelerine sahip görüntülerde zorluklarla karşılaşabilmekte ve bu durum yeniden yapılandırma için görüntülerin verimli şekilde seçilmesini engelleyebilmektedir.

Önerilen DSPCI-MH çerçevesi kapsamında ise DNN için dinamik bölme noktası belirleme sürecinde sağlanan önemli iyileştirmeleri ele alırken aynı zamanda giderilmesi gereken bazı sınırlamaları da vurgulamaktadır. Bu sınırlamalardan ilki, MH algoritmaların parametre ayarlarına duyarlılığı olmaktadır. MH algoritmalarının performansı büyük ölçüde popülasyon büyüklüğü, keşif-sömürü dengesi ve yakınsama kriterleri üzerinden hiperparametrelerin uygun şekilde seçilmesine bağlı olmaktadır. Bu parametreler, ağ koşulları, batch boyutu ve hesaplama kısıtlamalarına göre değişkenlik göstermesi sebebiyle farklı senaryolar için en uygun yapılandırmanın elde edilmesi için kapsamlı deneyler ve ince ayarlamalar gerektirmektedir. Parametrelerin manuel veya önceden tanımlanmış ayarlara bağımlı olması yöntemin farklı uygulamalara genellenebilirliğini azaltmakta ve öngörülemeyen koşullara uyumunu zorlaştırmaktadır. Bu bağlamda, gelecekteki çalışmalar, yöntem sağlamlığını artırmak amacıyla pekiştirmeli öğrenmeye dayalı veya kendini uyarlayan MH algoritmaları gibi adaptif parametre ayarlama tekniklerini keşfedebilir. Bir diğer sınırlama, gerçek zamanlı bölme noktası belirleme

sürecinin getirdiği hesaplama yükü olmaktadır. MH tabanlı yaklaşımlar çıkarım süresini azaltmada önemli avantajlar sağlasa da, optimizasyon süreci özellikle çok katmanlı ve büyük ölçekli DNN mimarileri için hesaplama açısından yoğun olabilmektedir. Bu tür DNN yapılarında optimal bölme noktasının belirlenmesi için gereken ek işlem süresi özellikle gerçek zamanlı kısıtların bulunduğu senaryolarda gecikmelere yol açabilmektedir. Ayrıca, ağ koşullarının sık değiştiği oldukça dinamik ortamlarda, karmaşık optimizasyon manzaraları uzun yakınsama sürelerine neden olabilmektedir. Bu zorlukların üstesinden gelmek için hibrit MH yaklaşımlarının kullanılması veya artımlı öğrenme mekanizmalarının entegrasyonu, bölme noktalarının geçmiş veriler ve sistem geri bildirimi temelinde dinamik olarak iyileştirilmesini sağlayarak çözüm sunabilir. Son olarak, önerilen yöntemin ölçeklenebilirliği, daha fazla araştırmaya açık bir alan olarak kalmaktadır. Önerilen çerçeve, orta ölçekli DNN mimarileri için bölme noktası seçimini başarılı bir şekilde optimize ederken transformatör tabanlı mimariler veya çok dallı ağlar gibi ultra büyük ölçekli DNN modeli yapılarına uygulanabilirliği daha fazla incelemeyi gerektirmektedir. Adaptif optimizasyon stratejileri, çok amaçlı ince ayarlama yöntemleri ve gelişmiş ölçeklenebilirlik mekanizmaları kullanılarak bu sınırlamaların ele alınması DSPCI-MH'nin çeşitli AI destekli IoT ve uç-bulut zekâ uygulamalarındaki uygulanabilirliğini artıracaktır.

SONUÇ

Bu tez çalışmasında, uç ve bulut bilişim kaynaklarının verimli kullanımı yoluyla IoT ortamlarında hesaplama süreçlerini geliştirmek için CI'dan yararlanmada yenilikçi bir yaklaşım olan DSPCI-MTL ve DSPCI-MH metodolojisini tanıtmaktadır. İlk olarak yenilikçi DSPCI-MTL yaklaşımı, görev yürütmeyi optimize etmek, işlem sürelerini önemli ölçüde iyileştirmek ve gerçek zamanlı veri ve ağ koşullarına dayalı en verimli bölme noktalarını dinamik olarak belirleyerek hesaplama yükünü azaltmak için MTL'i kullanmaktadır. Ayrıca, iletimler sırasında kaybolan verileri yeniden oluşturmak için AE'nin entegrasyonu ile yöntemin veri bütünlüğünü korumadaki ve değişen ağ senaryoları altında sistem performansını artırmadaki etkinliğinin altını çizmektedir. Genel olarak, DSPCI-MTL metodolojisi, IoT sistemlerindeki karmaşık hesaplama zorluklarını ele almak için MTL'i CI ile entegre etme potansiyelini vurgulamakla beraber aynı zamanda derin öğrenme teknolojilerinin pratik dağıtımında yeni bir ölçüt belirlemektedir. Verimli veri işleme ve işlemeyi sağlayarak kentsel analitiklerde veri bütünlüğünün, işleme verimliliğinin ve zamanında yürütmenin en önemli olduğu karmaşık IoT sistemlerine kadar çeşitli uygulamalarda gelecekteki gelişmeler için önemli vaat elde etmektedir. Ek olarak, CI mimarileri içindeki DNN'lerdeki bölünme noktalarını dinamik olarak belirlemek için önerilen DSPCI-MH çerçevesi olarak adlandırılan yeni bir yaklaşım sunulmaktadır. Önerilen bu diğer yenilikçi yöntem ise E2C ve E2E cihazlardaki dağıtılmış bilgi işlem ortamlarında performansı artırmak için GWO, PSO, ABC ve ARO gibi MH algoritmalarını entegre etmektedir. Birden fazla senaryodaki deneysel sonuçlar, MH tabanlı bölünme noktası belirlemesinin rastgele, matematiksel ve sezgisel tabanlı bölümlendirme yöntemleri dahil olmak üzere geleneksel yaklaşımlardan daha verimli performans gösterdiğini tutarlı bir şekilde göstermektedir. Bulgular, DSPCI-MH'nin çıkarım süresini önemli ölçüde azalttığını, enerji tüketimini optimize ettiğini ve bellek verimliliğini artırdığını vurgulayarak, onu gerçek zamanlı AI destekli IoT uygulamaları için etkili bir çözüm haline getirdiğini vurgulamaktadır. Uygulanan algoritmalar arasında PSO en iyi çıkarım verimliliğini gösterirken, ARO bellek kullanımını optimize ederek MH algoritmalarının dinamik ağ koşullarına ve hesaplama kısıtlamalarına uyarlanabilirliğini daha da güçlendirmektedir.

Avantajlarına rağmen, DSPCI-MH'nin performansı ağ koşulları, toplu boyut değişiklikleri ve hesaplama kısıtlamalarından etkilendiği için MH parametrelerinin ince ayarlanmasını gerektirmektedir. Bu zorlukların ele alınması, önerilen yaklaşımın sağlamlığını ve genelleştirilebilirliğini daha da artırması beklenmektedir. DSPCI-MTL ve DSPCI-MH teknikleri önemli ilerlemeler göstermekle beraber aynı zamanda iyileştirme alanlarını da ortaya çıkarmaktadır. Gelecekteki araştırmalar için, DSPCI-MTL ve DSPCI-MH tekniklerinin uyarlanabilirliği, ölçeklenebilirliği ve verimliliğinin çeşitli yönleri araştırılabilir:

• Sistemin gerçek zamanlı olarak, özellikle ultra düşük bant genişliği veya önemli ağ istikrarsızlığı koşulları altında uyum sağlama yeteneğini geliştirmek, otonom araçlar ve akıllı şehirler gibi kritik uygulamalar için önemini önemli ölçüde artıracaktır.

• DNN'nin karmaşık yapısının, optimum bölme noktasını belirlerken çıkarım süresini olumsuz etkilediği göz önüne alındığında, bu sorunu hafifletmek ve verimliliği artırmak için metasezgisel yaklaşımlar (Kiani vd., 2023) kullanılabilir.

• Üretken Çelişkili Ağlar (GAN'lar) (Seeliger vd., 2018) veya gelişmiş hibrit modeller gibi daha sofistike veri yeniden yapılandırma metodolojilerini araştırmak, önemli veri kaybı veya karmaşık görüntü veri kümeleriyle karakterize edilen bağlamlarda görüntü kurtarmanın dayanıklılığını artırabilir.

• Metodolojiyi, gerçek zamanlı video analitiği veya çevresel izleme gibi daha geniş bir IoT uygulamaları ve faaliyetleri yelpazesini kapsayacak şekilde değiştirmek, etkisini ve ölçeklenebilirliğini artırabilir.

• Hafif modeller ve sıkıştırma teknikleri geliştirmek, kaynak sınırlı uç cihazlardaki hesaplama yükünü hafifletecek ve dolayısıyla kısıtlı donanım yeteneklerine rağmen sorunsuz işlevselliği sağlayacaktır.

• Uç bulut etkileşimleri sırasında güvenli ve özel veri yönetimini garantilemek için şifreleme tekniklerini veya federasyon öğrenme metodolojilerini CI mimarisine dahil ederek gizlilik sorunlarını hafifletmek.

• Olası iyileştirmelerden biri, otomatik hiperparametre ayarlaması için pekiştirmeli öğrenme veya uyarlanabilir optimizasyon tekniklerini kullanarak manuel parametre ayarlamalarına olan ihtiyacı ortadan kaldırmaktır.

• DSPCI-MH, transformatör tabanlı mimari ve federasyon öğrenme senaryoları dahil olmak üzere ultra büyük derin öğrenme modelleri için ölçeklenebilir.

• Evrimsel algoritmaları derin takviyeli öğrenmeyle birleştiren hibrit MH stratejilerini dahil etmek, çerçevenin dalgalanan ağ koşullarına dinamik olarak uyum sağlama yeteneğini artırabilir.

KAYNAKÇA

Abd El Bar, W., Mahmoud, I. I., Konber, H. A., & Mongy, T. (2015). Image reconstruction technique using projection data from neutron tomography system. *Alexandria Engineering Journal*, *54*(4), 1057-1066. https://doi.org/10.1016/j.aej.2015.06.003

Ahmad, Z., Shahid Khan, A., Nisar, K., Haider, I., Hassan, R., Haque, M. R., Tarmizi, S., & Rodrigues, J. J. P. C. (2021). Anomaly Detection Using Deep Neural Network for IoT Architecture. *Applied Sciences*, *11*(15), 7050. https://doi.org/10.3390/app11157050

Ahuja, N., Datta, P., Kanzariya, B., Somayazulu, V. S., & Tickoo, O. (2023). Neural Rate Estimator and Unsupervised Learning for Efficient Distributed Image Analytics in Split-DNN models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022-2030. https://doi.org/10.1109/CVPR52729.2023.00201

Akay, B., Karaboga, D., & Akay, R. (2022). A comprehensive survey on optimizing deep learning models by metaheuristics. *Artificial Intelligence Review*, *55*(2), 829-894. https://doi.org/10.1007/s10462-021-09992-0

Aljabri, J., Alzaben, N., Nemri, N., Alahmari, S., Alotaibi, S. D., Alazwari, S., Khadidos, A. O., & Hilal, A. M. (2024). Hybrid stacked autoencoder with dwarf mongoose optimization for Phishing attack detection in internet of things environment. *Alexandria Engineering Journal*, *106*, 164-171. https://doi.org/10.1016/j.aej.2024.06.070

Alvar, S. R., & Bajic, I. V. (2019). Multi-Task Learning with Compressible Features for Collaborative Intelligence. *2019 IEEE International Conference on Image Processing (ICIP)*, 1705-1709. https://doi.org/10.1109/ICIP.2019.8803110

Alvar, S. R., & Bajic, I. V. (2020). Bit Allocation for Multi-Task Collaborative Intelligence. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4342-4346. https://doi.org/10.1109/ICASSP40776.2020.9054770

Alvar, S. R., Uyanik, K., & Bajic, I. V. (2022). License Plate Privacy in Collaborative Visual Analysis of Traffic Scenes. *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 300-305. https://doi.org/10.1109/MIPR54900.2022.00060

Anka, F., Agaoglu, N., Nematzadeh, S., Torkamanian-afshar, M., & Gharehchopogh, F. S. (2024). Advances in Artificial Rabbits Optimization: A Comprehensive Review. *Archives of Computational Methods in Engineering*. https://doi.org/10.1007/s11831-024-10202-7

Azadi, B., Haslgrübler, M., Anzengruber-Tanase, B., Sopidis, G., & Ferscha, A. (2024). Robust Feature Representation Using Multi-Task Learning for Human Activity Recognition. *Sensors*, *24*(2), 681. https://doi.org/10.3390/s24020681

Bajić, I. V. (2021). *Latent-Space Inpainting for Packet Loss Concealment in Collaborative Object Detection* (Versiyon 1). arXiv. https://doi.org/10.48550/ARXIV.2102.00142

Bajic, I. V., Lin, W., & Tian, Y. (2021). Collaborative Intelligence: Challenges and Opportunities. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8493-8497. https://doi.org/10.1109/ICASSP39728.2021.9413943

Bakhtiarnia, A., Milošević, N., Zhang, Q., Bajović, D., & Iosifidis, A. (2023). Dynamic Split Computing for Efficient Deep EDGE Intelligence. *ICASSP 2023 - 2023 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1-5. https://doi.org/10.1109/ICASSP49357.2023.10096914

Bertalmio, M., Bertozzi, A. L., & Sapiro, G. (2001). Navier-stokes, fluid dynamics, and image and video inpainting. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I-355-I-362. https://doi.org/10.1109/CVPR.2001.990497

Blanc-Durand, P., Schiratti, J.-B., Schutte, K., Jehanno, P., Herent, P., Pigneur, F., Lucidarme, O., Benaceur, Y., Sadate, A., Luciani, A., Ernst, O., Rouchaud, A., Creze, M., Dallongeville, A., Banaste, N., Cadi, M., Bousaid, I., Lassau, N., & Jegou, S. (2020). Abdominal musculature segmentation and surface prediction from CT using deep learning for sarcopenia assessment. *Diagnostic and Interventional Imaging*, *101*(12), 789-794. https://doi.org/10.1016/j.diii.2020.04.011

Bragilevsky, L., & Bajic, I. V. (2020). Tensor Completion Methods for Collaborative Intelligence. *IEEE Access*, *8*, 41162-41174. https://doi.org/10.1109/ACCESS.2020.2977050

Capogrosso, L., Cunico, F., Lora, M., Cristani, M., Fummi, F., & Quaglia, D. (2023). Split-Et-Impera: A Framework for the Design of Distributed Deep Learning Applications. *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 39-44. https://doi.org/10.1109/DDECS57882.2023.10139711

Chakraa, H., Guérin, F., Leclercq, E., & Lefebvre, D. (2023). Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, *168*, 104492. https://doi.org/10.1016/j.robot.2023.104492

Choi, H., & Bajic, I. V. (2018). Deep Feature Compression for Collaborative Object Detection. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 3743-3747. https://doi.org/10.1109/ICIP.2018.8451100

Cohen, R. A., Choi, H., & Bajic, I. V. (2021). Lightweight Compression of Intermediate Neural Network Features for Collaborative Intelligence. *IEEE Open Journal of Circuits and Systems*, *2*, 350-362. https://doi.org/10.1109/OJCAS.2021.3072884

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213-3223. https://doi.org/10.1109/CVPR.2016.350

Crawshaw, M. (2020). *Multi-Task Learning with Deep Neural Networks: A Survey* (Versiyon 1). arXiv. https://doi.org/10.48550/ARXIV.2009.09796

Cuong-Le, T., Minh, H.-L., Sang-To, T., Khatir, S., Mirjalili, S., & Abdel Wahab, M. (2022). A novel version of grey wolf optimizer based on a balance function and its application for hyperparameters optimization in deep neural network (DNN) for structural damage identification. *Engineering Failure Analysis*, *142*, 106829. https://doi.org/10.1016/j.engfailanal.2022.106829

Danba, S., Bao, J., Han, G., Guleng, S., & Wu, C. (2022). Toward Collaborative Intelligence in IoV Systems: Recent Advances and Open Issues. *Sensors*, *22*(18), 6995. https://doi.org/10.3390/s22186995

Darwish, A., Hassanien, A. E., & Das, S. (2020). A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review*, *53*(3), 1767-1812. https://doi.org/10.1007/s10462-019-09719-2

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. https://doi.org/10.1109/CVPR.2009.5206848

Dhondea, A., Cohen, R. A., & Bajic, I. V. (2021). CALTEC: Content-Adaptive Linear Tensor Completion For Collaborative Intelligence. *2021 IEEE International Conference on Image Processing (ICIP)*, 2179-2183. https://doi.org/10.1109/ICIP42928.2021.9506372

Duan, Y., & Wu, J. (2021). Joint Optimization of DNN Partition and Scheduling for Mobile Cloud Computing. *50th International Conference on Parallel Processing*, 1-10. https://doi.org/10.1145/3472456.3472468

Eddahmani, I., Pham, C.-H., Napoléon, T., Badoc, I., Fouefack, J.-R., & El-Bouz, M. (2023). Unsupervised Learning of Disentangled Representation via Auto-Encoding: A Survey. *Sensors*, *23*(4), 2362. https://doi.org/10.3390/s23042362

Eshratifar, A. E., Abrishami, M. S., & Pedram, M. (2021). JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services. *IEEE Transactions on Mobile Computing*, *20*(2), 565-576. https://doi.org/10.1109/TMC.2019.2947893

Gao, M., Cui, W., Gao, D., Shen, R., Li, J., & Zhou, Y. (2019). Deep Neural Network Task Partitioning and Offloading for Mobile Edge Computing. *2019 IEEE Global Communications Conference (GLOBECOM)*, 1-6. https://doi.org/10.1109/GLOBECOM38437.2019.9013404

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* (Versiyon 1). arXiv. https://doi.org/10.48550/ARXIV.1704.04861

Hu, S., Deng, R., Du, X., Lu, Z., Duan, Q., He, Y., Huang, S.-C., & Wu, J. (2024). *LAECIPS: Large Vision Model Assisted Adaptive Edge-Cloud Collaboration for IoT-based Perception System* (Versiyon 1). arXiv. https://doi.org/10.48550/ARXIV.2404.10498

Jahier Pagliari, D., Chiaro, R., Macii, E., & Poncino, M. (2020). CRIME: Input-Dependent Collaborative Inference for Recurrent Neural Networks. *IEEE Transactions on Computers*, 1-1. https://doi.org/10.1109/TC.2020.3021199

Jankowski, M., Gunduz, D., & Mikolajczyk, K. (2020). Joint Device-Edge Inference over Wireless Links with Pruning. *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 1-5. https://doi.org/10.1109/SPAWC48557.2020.9154306

Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., & Tang, L. (2017). Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 615-629. https://doi.org/10.1145/3037697.3037698

Karjee, J., Anand, K., Bhargav, V. N., Naik, P. S., Dabbiru, R. B. V., & Srinidhi, N. (2021). Split Computing: Dynamic Partitioning and Reliable Communications in IoT-Edge for 6G Vision. *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, 233-240. https://doi.org/10.1109/FiCloud49777.2021.00041

Karjee, J., Anand, K., S, P. N., Dabbiru, R. B. V., Byadgi, C. S., & N, S. (2022). Dynamic Split Computing of PoseNet Inference for Fitness Applications in Home IoT-Edge Platform. *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 430-432. https://doi.org/10.1109/COMSNETS53615.2022.9668605

Karjee, J., Naik S, P., Anand, K., & Bhargav, V. N. (2022). Split computing: DNN inference partition with load balancing in IoT-edge platform for beyond 5G. *Measurement: Sensors*, *23*, 100409. https://doi.org/10.1016/j.measen.2022.100409

Kaya, E., Gorkemli, B., Akay, B., & Karaboga, D. (2022). A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. *Engineering Applications of Artificial Intelligence*, *115*, 105311. https://doi.org/10.1016/j.engappai.2022.105311

Kiani, F., Nematzadeh, S., Anka, F. A., & Findikli, M. A. (2023). Chaotic Sand Cat Swarm Optimization. *Mathematics*, *11*(10), 2340. https://doi.org/10.3390/math11102340

Lee, J., Kang, S., Lee, J., Shin, D., Han, D., & Yoo, H.-J. (2020). The Hardware and Algorithm Co-Design for Energy-Efficient DNN Processor on Edge/Mobile Devices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *67*(10), 3458-3470. https://doi.org/10.1109/TCSI.2020.3021397

Li, E., Zeng, L., Zhou, Z., & Chen, X. (2020). Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Transactions on Wireless Communications*, *19*(1), 447-457. https://doi.org/10.1109/TWC.2019.2946140

Liu, J., Musialski, P., Wonka, P., & Ye, J. (2013). Tensor Completion for Estimating Missing Values in Visual Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 208-220. https://doi.org/10.1109/TPAMI.2012.39

Mao, J., Yang, Z., Wen, W., Wu, C., Song, L., Nixon, K. W., Chen, X., Li, H., & Chen, Y. (2017). MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs. *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 751-756. https://doi.org/10.1109/ICCAD.2017.8203852

Matsubara, Y., Levorato, M., & Restuccia, F. (2023). Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Computing Surveys*, *55*(5), 1-30. https://doi.org/10.1145/3527155

Miao, W., Zeng, Z., Wei, L., Li, S., Jiang, C., & Zhang, Z. (2020). Adaptive DNN Partition in Edge Computing Environments. *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 685-690. https://doi.org/10.1109/ICPADS51040.2020.00097

Mrudula, S. T., Meenakshi, Ritonga, M., Sivakumar, S., Jawarneh, M., F, S., Keerthika, T., Rane, K. P., & Roy, B. (2024). Internet of things and optimized knn based intelligent transportation system for traffic flow prediction in smart cities. *Measurement: Sensors*, *35*, 101297. https://doi.org/10.1016/j.measen.2024.101297

Nadimi-Shahraki, M. H., Zamani, H., Asghari Varzaneh, Z., Sadiq, A. S., & Mirjalili, S. (2024). A systematic review of applying grey wolf optimizer, its variants, and its developments in different Internet of Things applications. *Internet of Things*, *26*, 101135. https://doi.org/10.1016/j.iot.2024.101135

Nilsson, J., & Akenine-Möller, T. (2020). *Understanding SSIM* (Versiyon 2). arXiv. https://doi.org/10.48550/ARXIV.2006.13846

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context Encoders: Feature Learning by Inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2536-2544. https://doi.org/10.1109/CVPR.2016.278

Pham, T. (2021). Semantic Road Segmentation using Deep Learning. *2020 Applying New Technology in Green Buildings (ATiGB)*, 45-48. https://doi.org/10.1109/ATiGB50996.2021.9423307

Riaz, M., Bashir, M., & Younas, I. (2022). Metaheuristics based COVID-19 detection using medical images: A review. *Computers in Biology and Medicine*, *144*, 105344. https://doi.org/10.1016/j.compbiomed.2022.105344

Rodriguez-Conde, I., Campos, C., & Fdez-Riverola, F. (2023). Horizontally Distributed Inference of Deep Neural Networks for AI-Enabled IoT. *Sensors*, *23*(4), 1911. https://doi.org/10.3390/s23041911

Sbai, M., Saputra, M. R. U., Trigoni, N., & Markham, A. (2021). Cut, Distil and Encode (CDE): Split Cloud-Edge Deep Inference. *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 1-9. https://doi.org/10.1109/SECON52354.2021.9491600

Seeliger, K., Güçlü, U., Ambrogioni, L., Güçlütürk, Y., & Van Gerven, M. A. J. (2018). Generative adversarial networks for reconstructing natural images from brain activity. *NeuroImage*, *181*, 775-785. https://doi.org/10.1016/j.neuroimage.2018.07.043

Seyyedabbasi, A., Kiani, F., Allahviranloo, T., Fernandez-Gamiz, U., & Noeiaghdam, S. (2023). Optimal data transmission and pathfinding for WSN and decentralized IoT systems using I-GWO and Ex-GWO algorithms. *Alexandria Engineering Journal*, *63*, 339-357. https://doi.org/10.1016/j.aej.2022.08.009

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, *10*, 10031-10061. https://doi.org/10.1109/ACCESS.2022.3142859

Shiranthika, C., Saeedi, P., & Bajić, I. V. (2023). Decentralized Learning in Healthcare: A Review of Emerging Techniques. *IEEE Access*, *11*, 54188-54209. https://doi.org/10.1109/ACCESS.2023.3281832

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (Versiyon 6). arXiv. https://doi.org/10.48550/ARXIV.1409.1556

Singh, R., & Gill, S. S. (2023). Edge AI: A survey. *Internet of Things and Cyber-Physical Systems*, *3*, 71-92. https://doi.org/10.1016/j.iotcps.2023.02.004

Tang, X., Chen, X., Zeng, L., Yu, S., & Chen, L. (2021). Joint Multiuser DNN Partitioning and Computational Resource Allocation for Collaborative Edge Intelligence. *IEEE Internet of Things Journal*, *8*(12), 9511-9522. https://doi.org/10.1109/JIOT.2020.3010258

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2023). *Loss Functions and Metrics in Deep Learning* (Versiyon 4). arXiv. https://doi.org/10.48550/ARXIV.2307.02694

Thakkar, A., & Lohiya, R. (2023). Fusion of statistical importance for feature selection in Deep Neural Network-based Intrusion Detection System. *Information Fusion*, *90*, 353-363. https://doi.org/10.1016/j.inffus.2022.09.026

Tomar, V., Bansal, M., & Singh, P. (2024). Metaheuristic Algorithms for Optimization: A Brief Review. *RAiSE-2023*, 238. https://doi.org/10.3390/engproc2023059238

Uyanik, K., Yeganli, S. F., & Bajić, I. V. (2023). Grad-FEC: Unequal Loss Protection of Deep Features in Collaborative Intelligence. *2023 IEEE International Conference on Image Processing (ICIP)*, 3140-3144. https://doi.org/10.1109/ICIP49359.2023.10222115

Wang, H., Cai, G., Huang, Z., & Dong, F. (2019). ADDA: Adaptive Distributed DNN Inference Acceleration in Edge Computing Environment. *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 438-445. https://doi.org/10.1109/ICPADS47876.2019.00069

Wang, X., Wang, Y., Javaheri, Z., Almutairi, L., Moghadamnejad, N., & Younes, O. S. (2023). Federated deep learning for anomaly detection in the internet of things. *Computers and Electrical Engineering*, *108*, 108651. https://doi.org/10.1016/j.compeleceng.2023.108651

Wang, Y., Wang, W., Liu, D., Jin, X., Jiang, J., & Chen, K. (2021). Enabling Edge-Cloud Video Analytics for Robotics Applications. *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 1-10. https://doi.org/10.1109/INFOCOM42981.2021.9488801

Wang, Z., Wang, E., & Zhu, Y. (2020). Image segmentation evaluation: A survey of methods. *Artificial Intelligence Review*, *53*(8), 5637-5674. https://doi.org/10.1007/s10462-020-09830-9

Xiang, H., Zou, Q., Nawaz, M. A., Huang, X., Zhang, F., & Yu, H. (2023). Deep learning for image inpainting: A survey. *Pattern Recognition*, *134*, 109046. https://doi.org/10.1016/j.patcog.2022.109046

Xu, D., He, X., Su, T., & Wang, Z. (2023). *A Survey on Deep Neural Network Partition over Cloud, Edge and End Devices* (Versiyon 1). arXiv. https://doi.org/10.48550/ARXIV.2304.10020

Yao, S., Li, J., Liu, D., Wang, T., Liu, S., Shao, H., & Abdelzaher, T. (2020). Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 476-488. https://doi.org/10.1145/3384419.3430898

Yatbaz, H. Y., Dianati, M., & Woodman, R. (2024). Introspection of DNN-Based Perception Functions in Automated Driving Systems: State-of-the-Art and Open Research

Challenges. *IEEE Transactions on Intelligent Transportation Systems*, *25*(2), 1112-1130. https://doi.org/10.1109/TITS.2023.3315070

Yu, D., Du, X., Jiang, L., Tong, W., & Deng, S. (2024). EC-SNN: Splitting Deep Spiking Neural Networks for Edge Devices. *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*, 5389-5397. https://doi.org/10.24963/ijcai.2024/596

Zeng, L., Chen, X., Zhou, Z., Yang, L., & Zhang, J. (2021). CoEdge: Cooperative DNN Inference With Adaptive Workload Partitioning Over Heterogeneous Edge Devices. *IEEE/ACM Transactions on Networking*, *29*(2), 595-608. https://doi.org/10.1109/TNET.2020.3042320

Zhang, Y., & Yang, Q. (2018). An overview of multi-task learning. *National Science Review*, *5*(1), 30-43. https://doi.org/10.1093/nsr/nwx105

Zhang, Y., & Yang, Q. (2022). A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering*, *34*(12), 5586-5609. https://doi.org/10.1109/TKDE.2021.3070203

Zhang, Z., Wang, M., Ma, M., Li, J., & Fan, X. (2021). MSFC: Deep Feature Compression in Multi-Task Network. *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 1-6. https://doi.org/10.1109/ICME51207.2021.9428258

Zhao, Y., Wang, X., Che, T., Bao, G., & Li, S. (2023). Multi-task deep learning for medical image computing and analysis: A review. *Computers in Biology and Medicine*, *153*, 106496. https://doi.org/10.1016/j.compbiomed.2022.106496

Zhou, L., Song, R., Chen, G., Festag, A., & Knoll, A. (2023). Residual encoding framework to compress DNN parameters for fast transfer. *Knowledge-Based Systems*, *277*, 110815. https://doi.org/10.1016/j.knosys.2023.110815