Avrupa Bilim ve Teknoloji Dergisi
Sayı 33, S. 10-19, Ocak 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 33, pp. 10-19, January 2022

Research Article

Derin Öğrenme Algoritmalarının GPU ve CPU Donanım Mimarileri Üzerinde Uygulanması ve Performans Analizi: Deneysel Araştırma

Tuğba Saray Çetinkaya[1*], Ahmet Sertbaş[2]

[1*] İstanbul Gelişim Üniversitesi, İstanbul Meslek Yüksek Okulu, Bilişim Güvenliği Teknolojisi Programı, İstanbul, Türkiye, (ORCID: 0000-0003-1639-553X), tsaray@gelisim.edu.tr

[2] İstanbul Üniversitesi - Cerrahpaşa, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye (ORCID: 0000-0001-8166-1211), asertbas@iuc.edu.tr

(İlk Geliş Tarihi 20 Mayıs 2021 ve Kabul Tarihi 19 Ocak 2022)

(**DOI:** 10.31590/ejosat.937936)

ATIF/REFERENCE: Çetinkaya Saray, T. & Sertbaş, A. (2022). Derin Öğrenme Algoritmalarının GPU ve CPU Donanım Mimarileri Üzerinde Uygulanması ve Performans Analizi: Deneysel Araştırma. *Avrupa Bilim ve Teknoloji Dergisi,* (33), 10-19.

Öz

Günümüzde hızla gelişen teknolojiyle verilerin çeşitliliği ve boyutu artmaktadır. Bu artış bilgisayar mimarisinde farklı tasarımları ortaya çıkarmıştır. CPU ve GPU mimarileri üzerlerinde bulunan çekirdek sayıları uygulama anında sonuca ulaşmada çözümler sağlayabilmektedir. Yazılım geliştirmesi yapılırken işlem performansı ve güç tüketimine dikkat edilmelidir. CPU'lar GPU'lardan daha uzun işlem süresi ile uygulamaları yürütmektedir. Bu süre performans sırasında harcanan gücü doğru orantılı etkilemektedir. GPU'lar derin öğrenme algoritmalarında CPU'lardan daha hızlı ve başarılı sonuçlar vermektedir. Öğrenme aşamasındaki en önemli kriter olan veri setinin büyüklüğü ve çeşitliliği öğrenme başarısını aynı oranda artırmaktadır. Bu çalışmada farklı mimariye sahip işlemciler üzerinde veri seti büyüklüğü ve işlem süresi kriterleri göz önünde bulundurularak uygulamalar yapılmıştır. Yapılan uygulamalarda GPU mimarilerinde harcanan güç seviyesi ölçülmüştür. Farklı büyüklüğe sahip 3 veri seti üzerinde CNN, RNN ve LSTM derin öğrenme algoritmaları uygulanmıştır. 6 farklı deney yapılarak performans ve enerji tüketimi konularında tespitler ve performans karşılaştırılması yapılmıştır. Çalışma neticesinde elde edilen sonuçlar ile algoritmalar üzerinde çalışmalar yapılırken süre ve enerji kriterleri baz alınmıştır. Bulgular derin öğrenme algoritmalarının yüksek doğrulukta GPU sistemlerinde tahmin edilmesinde yardımcı bir araç olarak kullanılabileceği yönündedir. Araştırmanın sonuçları CPU ve GPU sistemleri ile enerji ve süre açısından önemli bilgiler içermesinin yanı sıra, gelecekte farklı sektörlerde uygulanması açısından değer taşımaktadır.

Anahtar Kelimeler: Derin Öğrenme, GPU, CPU, Bilgisayar Mimarisi, Performans Analizi.

Application of Deep Learning Algorithms to GPU and CPU Hardware Architectures and Performance Analysis: Experimental Reseach

Abstract

Nowadays, the variety and volume of data is increasing with the rapidly developing technology. This increase has led to different designs in computer architecture. The number of cores on CPU and GPU architectures can provide solutions to achieve the result at the time of application. When developing software, attention should be paid to processing performance and power consumption. CPUs run applications with a longer processing time than GPUs. This time directly affects power consumption during performance. GPUs produce faster and more successful results than CPUs for Deep Learning algorithms. The size and diversity of the dataset, which is the most important criterion in the learning phase, increases the learning success to the same extent. In this study, applications were performed on processors with different architectures considering the criteria of dataset size and processing time. In the applications, the power consumption of GPU architectures is measured. CNN, RNN and LSTM deep learning algorithms are

[*] Sorumlu Yazar: tsaray@gelisim.edu.tr

applied to 3 different sized datasets. 6 different experiments are performed and determinations and performance comparisons are made on the performance and power consumption. Time and energy criteria were used in processing the algorithms with the results of the study. The results show that Deep Learning algorithms can be used as a tool for predicting GPU systems with high accuracy. The results of the study not only contain important information related to CPU and GPU systems, energy, and time, but also are valuable for future applications in various fields.

Keywords: Deep learning, GPU, CPU, Computer Architecture, Performance Analysis.

1. Giriş

Günümüz bilgisayar uygulamalarında yüksek hızlı ve düşük güçlü hesaplama ihtiyacı gittikçe artan ve çözümü zor bir problemdir. Son yıllarda, yeni bilgisayar mimarilerinin geliştirilmesinde güç verimliliği ve yüksek performans özellikleri baskın kriterler haline gelmiştir. Bu kapsamda, güç verimliliği açışından mevcut ihtiyaçlara cevap veren yeni sistemler planlanırken, artan tasarım zorluğuna çözüm olarak sistemi parça düzeyinde ele alınmasına odaklanılmaktadır. Bu yaklaşım da paralellik ve farklı yapılara dayanan tasarımlar öne çıkarmaktadır.

Bu makale şu şekilde düzenlenmiştir: Çalışmanın ikinci bölümünde yapay zeka alanında uygulamalar ve güç analizi konusunda literatürde yer alan çalışmalar incelenmiştir. Makalenin üçüncü bölümünde çalışmanın metodolojisi, dördüncü bölümde ise yapılan deneylerden elde edilen görsel çıktılar ve test sonuçları verilmiştir. Son bölümde ise, çalışmada elde edilen sonuçlar değerlendirilmiştir.

2. İlgili Çalışmalar

Bilgisayar sistemlerinde devre performansını arttıran, maliyeti ve güç harcama seviyesini düşüren teknolojiler üzerinde durulmaktadır. Gordon Moore'un (Moore Yasası) teknoloji ölçeklendirme tahminleri, üretim entegrasyonunu (yonga başına transistör) ölçeklendirerek üretim maliyetine odaklanmıştır. Doğal olarak daha sonraki tasarımlar uzun yıllar boyunca bu maliyet odaklı performans iyileştirmeleri üzerine olmuştur. Dennard ise transistörlerin boyutu küçüldüğünde, besleme voltajlarının düşmesi üzerine bir ölçeklendirme ilkesi ortaya koymuştur. Bu ölçeklendirme, Moore Yasası tarafından ortaya konulmuş olan transistör artışının etkisine dair görüşünün, performans iyileştirmesi ve güç tasarrufu olarak ikiye ayrılmasına neden olmuştur. Daha sonra bu yöntemler, sistemlerin gücü verimli bir şekilde ve daha uzun süre kullanması için yetersiz kaldığında komut düzeyinde paralellik (ILP- instruction-level parallelism) ile bilgisayar mimarisinde alternatif paralellik türleri (görev / iş parçacığında paralellik gibi) keşfedilmeye başlanmıştır. Bu sayede çok çekirdekli (Multicore) mimari tasarlanmıştır (Själander ve ark., 2014). Tek işlem çoklu veri (SIMD) mimarisi, performansı artırmak için yaygın olarak kullanılan ve etkin performansa sahip bir paralel işleme yöntemdir. Çok çekirdekli sistemlerde kullanılmaktadır. Birim zamanda tek bir işlem gerçekleştirilmektedir (Huang ve ark., 2019) (Mumcu ve ark., 2020).

Yapay zeka alanında gerçekleştirilen bir çok örnek gerçek yaşamdan temelini almaktadır (Nabiyev, 2003). Yapay zeka yöntemlerinden bulanık mantık, yapay sinir ağları, destek vektör makineleri ve rastgele ormanlar kullanılarak hayvan cinsiyetinin tahmini konusunda yüksek tahminleme çalışması gerçekleştirilmiştir (Öztürk ve ark., 2022). Yüz tanıma sistemlerinde kullanılmak üzere Haar-Cascades sınıflandırıcısı ve LBPH (Yerel İkili Desenler Histogramları) kullanılarak diğer yüz tanıma sistemlerinden farklı bir sistem geliştirilmiştir (Karadağ ve ark., 2020). Derin öğrenme teknikleri kullanılarak geliştirilen Faster R-CNN (Faster Region Based Convolutional Networks) evrişimli sinir ağı ile geliştirilen çalışma içerisinde "Bardak" veri seti kullanılarak yüksek nesne tanıma başarısı elde edilmiştir (Yılmaz ve ark., 2020). Uykuda solunum bozukluğu konusunda makine öğrenmesi teknikleri ile uykuda solunum bozukluğu türleri sınıflandırılmıştır (Balci, 2022). Tıbbi görüntüleme tabanlı hastalıkların teşhis etmek için evrişimli sinir ağı (CNN) tabanlı bir tıbbı görüntü birleştirme algoritması geliştirilmiştir (Abas ve ark., 2021). Geliştirilen bir çalışmada sosyal medyada ifadelerini üzerinde doğal dil işleme teknikleriyle duyguya dayalı anahtar kelimeler içeren Türkçe sözlük oluşturularak sosyal medya duyarlılık analizi yapılmıştır (Uysal ve ark., 2017). Geliştirilen bir çalışmada havuçların fiziksel özelliklere göre hacimlerini tahmin etmek için derin öğrenme yaklaşımlarından LSTM (Uzun Kısa Süreli Bellek) ve DFN (Derin İleri Beslemeli Ağ) algoritmaları kullanılarak yüksek tahmin oranı elde edilmiştir (Örnek ve ark., 2021). Asma yaprağı sınıflandırması için derin öğrenmeye dayalı bir sınıflandırma çalışması ile öznitelik seçiminin sınıflandırma başarısını arttırdığı ölçülmüştür (Koklu ve ark., 2021). Yangının tespitinden derin öğrenme yaklaşımıyla dumanın renk histogramını kullanarak yangın ve duman algılama algoritması geliştirilmiştir (Lee ve ark., 2019).

Son yıllarda ise güç kısıtlamalarının basit modül seviyesindeki kontrollerle yönetilmesi zorlaştıkça, daha iyi güç çözümleri sunabilecek tasarımlara odaklanılmıştır. Özellikle farklı güç ve performans özelliklerine sahip sistemlerin çeşitliliğini sağlamak için yonga işleme modülleri farklılaştırılmıştır. Bu farklılaşmalar homojen veya heterojen paralellik özelliğine sahip yonga tasarımları ile gerçekleştirilmektedir (Själander ve ark., 2014). Bilgisayar mimarisi içerisinde işlemciler, güç kullanımı açısından büyük orana sahiptir. Özellikle veri taşıma ve veri depolama gibi işlemler güç sorunlarına neden olmaktadır. Bilgisayar sistemlerinde, gerçek aritmetik birimler üzerinde görüntü işleme gibi kapasiteyi yoğun olarak kullanan işlem gereksinimleri arttıkça gelişen bu sistemlerin güç tüketimi de artmaktadır. Böyle durumlarda gömülü işlemciler masaüstü veya sunucu işlemcilerinden çok daha az güç kullanmaktadır, ancak kayan noktalı aritmetik işlemler için sınırlı destek sağlamakta veya hiç destek vermemektedir. İşlem kapasitesini arttırmak için daha karmaşık yapıda sistemler geliştirilmektedir (Själander ve ark., 2014) (Huynh ve ark., 2012). Bu yapılardan biri olan karma (Hybrid) yeniden yapılandırılabilir işlemciler, daha iyi komut yürütme performansı ve güç tüketimini dengelemek için sabit ve yeniden yapılandırılabilir bilgi işlem yapılarını birleştirmektedirler (Huynh ve ark., 2012). Çekirdek sayısı artmış GPU mimariler ve kriptografik motorlar gibi hızlandırıcılarla da birleştirilebilmektedir (Själander ve ark., 2014). Ayrıca yazılım ile programlanabilen donanımlar da (FPGA) yüksek kapasiteli işlemlerde düşük maliyet ve düşük güç tüketimi ihtiyacını sağlamak için kullanılmakta ve amaca

göre optimize edilebilmektedir (Dehnavi ve ark., 2018) (Kahoul ve ark., 2009).

Düşük güçte herhangi bir sistemi tasarlamak için, sistemin farklı seviyelerde güç optimizasyonuna ihtiyaç duyulmaktadır. Sistem düzeyindeki mimarilerinin çoğu sıralı devrelerden oluşmakta, bu devrelerin tasarımı sistemin genel gücünü azaltmada önemli bir rol oynamaktadır (Katreepalli ve ark., 2019) (Sanapala, 2017). Etkin güç tüketim kavramı, içerisinde mikro denetleyici birimi (MCU- microcontroller unit) içeren akıllı mobil cihazlar için de büyük öneme sahiptir. MCU, gerçek zamanlı algılama görevleri gerçekleştirirken olağanüstü düşük güç performansına sahip olmalarını sağlayan farklı çalışma modları arasında dinamik olarak geçiş yapabilir. Donanım optimizasyonunun yanı sıra, bir MCU ile enerji verimliliğini ve hizmet kalitesini dengelemek için programlama algoritmalarıyla farklı tasarımlar gerçekleştirilmektedir (Lautner ve ark., 2018).

Günümüzde belirli görevleri yerine getirmek için belirli bir amaca özel yüksek performans ve hacim verimliliği sunan mimariler geliştirilebilmektedir. Bu mimariler sayesinde çoklu işlem protokolleri, ayrı modüller ile kontrol edilebilmektedir (Li ve ark., 2012) (Veera ve ark., 2019) (Sriadibhatla ve ark., 2019). Makine öğrenmesi, görüntü işleme, derin öğrenme gibi yüksek performans gerektiren uygulamalar geliştirmek için CPU, GPU, FPGA ve TPU platformlarından oluşan farklı mimarilere sahip seçenekler mevcuttur. En iyi performansı ve düşük maliyet için hangi platformu seçmek büyük önem taşımaktadır (Çetin ve ark., 2013) (InAccel, 2018) (Goz ve ark., 2020) (Demirbas ve ark., 2020). İşlemci mimarilerin çalışırken bilgi işlem performansları ölçülebilmektedir (Wyant ve ark., 2012) (Mittal ve ark., 2014).

FPGA ve GPU hızlandırıcıları içeren hibrit sistemler önemli performans iyileştirmeleri sağlayabilirken aynı zamanda verimliliği artırabilmektedir. Ayrıca CUDA sistemi, GPU'ları programlamak için yüksek verimlilik yaklaşımı sağlamaktadır (Betkaoui ve ark., 2010) (Holm ve ark., 2020) (Vaidya, 2018). GPU mimariler, yüksek performanslı işlemler için paralel hesaplama çözümleri sunabilmektedirler (Bandyopadhyay, 2019). Yüksek performanslı gömülü görüntü uygulamaları geliştirmek için çalışma zamanı ve performansı enerji kısıtlamalarıyla dengelenmelidir (Qasaimeh ve ark., 2019).

Tablo 1. Literatür taraması kapsamında değerlendirilen çalışmalar.

Çalışma İsmi	Mimari Tipi	Gerçekleştiriken Uygulama Tipi
Huynh ve ark., 2012	CPU	Güç verimli hesaplama
Lautner ve ark., 2018	Mikrokontroller	Güç verimliliği algoritmaları
Wyant ve ark., 2012	CPU, GPU ve FPGA	Performans karşılaştırılması
Mumcu ve ark., 2020	GPU	Cuda üzerinde paralel programlama
Holm ve ark., 2020	GPU	Python ile performans, güç ve verimlilik analizi
Aydın ve ark., 2020	GPU	Cuda üzerinde paralel programlama

3. İşlemci Mimarileri ve Kullanılan Derin Öğrenme Yöntemleri

Çalışmanın metodolojisi; işlemci mimarileri üzerinde geliştirilen yazılımların donanımlar üzerinde işlem performansı ve güç tüketimi kavramlarının genel çerçevesini çizmek, güç analizini araştırmak, bu çerçevede GPU (Graphics Processing Unit - Grafik İşlemci Ünitesi) ve CPU (Central Processing Unit – Merkezi İşlem Birimi)'nin mimarilerini incelemek, bu mimariler üzerinde geliştirilen algoritmaları tespit etmek ve algoritmaların mimariler üzerinde güç ve süre tüketimini önemini ortaya koymak ve öneriler getirmektir. Bu kapsamda nicel bir araştırma yöntemi kullanılarak araştırma yapılmıştır. Araştırmanın konusu ve amacı doğrultusunda belirlenen sorulara yanıtlar aranmıştır:

- CPU ve GPU mimari sistemlerinde güç analizi, cuda, güç tüketimi, güç kaybı ve kullanılan algoritmaların genel çerçevesi neleri içermektedir?

- Güç analizine yönelik geçmişte yapılmış çalışmalar nasıl değerlendirilmiştir?

- Yapay Zeka algoritmalarının CPU ve GPU mimarilerinde yapılmış çalışmaları hangileridir?

- Mimarilerde kullanılacak algoritmalardan elde edilen sonuçlar kapsamında hangi öneriler getirilebilir?

Araştırma kapsamında özellikle CPU ve GPU mimarilerinde güç analizi konusunda var olan durumlar ve bu durumlardan elde edilen sonuçlar kapsamında öneriler getirilmeye çalışılmıştır. Çalışmada kapsamında, Türkiye ve yurtdışı literatürde olan kaynaklar taranmış, bu kaynaklardan elde edilen sonuçlar karşılaştırılarak ve çalışma neticesinde ortaya çıkan sonuçlar ele alınarak işlemci mimarileri konusunda vurgu yapılmıştır.

3.1. CPU Hesaplama

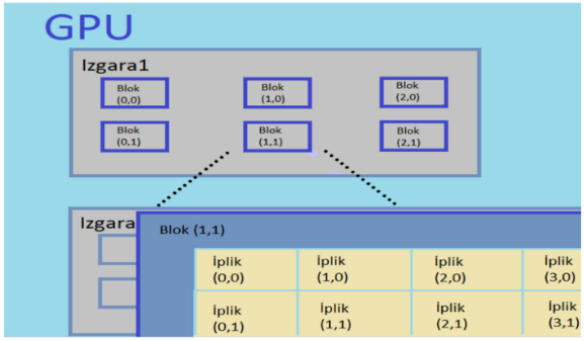
CPU bir diğer adıyla işlemci, milyarlarca mikroskobik boyutta transistörden oluşan küçük boyutta bir donanımdır. Aritmetik mantık birimi (ALU) ve kontrol birimi (CU) olmak üzere iki ana bileşenden oluşmaktadır. CPU temel olarak talimatları almak, çözmek, yürütmek ve saklamak işlevlerini sırasıyla yerine getirmektedir. Bir saniye içerisinde bu sırada gerçekleştirdiği döngülerin sayısı işlemci hızını oluşturmaktadır. Teknolojinin gelişmesiyle temel mimariler kullanılarak birçok faklı tasarım oluşmaktadır.

CPU'nun çok çeşitli görevleri hızlı işleyecek şekilde tasarlanması, ancak çalışabilecek görevlerin eşzamanlılığıyla sınırlı olmasıdır. CPU çalışma performansı üzerinde yürütülen görevlerin sayısından, kesmelerden ve görev anahtarlarının sıklığından doğrudan etkilenmektedir (Dodiu ve ark., 2012). CPU mimarisinde bir program derlendiğinde harcanan gücün büyük bir bölümünü işlemci kullanmaktadır. Bu değer tüketilen enerjinin ortalama olarak %88.94' ünü temsil etmektedir. Geriye kalan bölüm DRAM tarafından kullanılmaktadır (Pereira ve ark., 2017).

3.2. GPU Mimarisi

GPU, yüksek çözünürlüklü görüntüleri ve videoları aynı anda hızlı bir şekilde işlemek için tasarlanmıştır (Stratton ve ark., 2012). GPU'lar birden çok veri kümesi üzerinde paralel işlemler gerçekleştirebildiğinden, makine öğrenimi ve bilimsel hesaplama gibi görüntü işleme uygulamaları dışında kalan görevler için de yaygın olarak kullanılmaktadır. Yüksek performanslı hesaplama kodları üzerinde gerçekleştirilen uygulamalarda Python kullanmanın etkisi ihmal edilebilir düzeydedir ve eşdeğer seviyeye ayarlanmış CUDA ve OpenCL uygulamalarında çoğu durumda aynı hesaplama performansı elde edilebilmektedir (Holm ve ark., 2020) (Aydın ve ark., 2020).

GPU özellikle grafik hesaplama yapmak için tasarlanmış bir işlemci mimarisidir. CPU mimarisinin yapısına göre daha fazla CU ve ALU birimine sahiptir. Ayrıca komutlar paralel olarak işlenmektedir. Bu sayede grafik hesaplama işlemlerinde daha verimli ve hızlı performansa sahiptir. GPU programlama dilleri, kavramsal olarak birbirine çok benzerdir ve aynı tür paralellik ilkelerine sahiptir. GPU teknolojisi, programlama dilleri ve kütüphaneler aracılığıyla GPU üzerinde tam erişimli çalışabilmektedir. Bu bölümde donanımsal olarak GPU özelliklerinden ve türlerinden bahsedilmiştir. GPU mimari yapısı Şekil 1 üzerinde verilmiştir.



Şekil 1. GPU mimari yapısı.

GPU çekirdeğinde şekil 1 üzerinde de görülen iş parçacığı (iplik), blok ve ızgara birimleri mevcuttur. Bu birimler GPU hesaplama işlemlerinde mimariden verimli şekilde yararlanmamızı sağlamaktadır. GPU için oluşturulan CUDA (Compute Unified Device Architecture) yazılım eklentisi ile birlikte yeni bir mimari ortaya çıkmıştır. Bir CUDA programı, bir grup iş parçası bloğundaki birden fazla iş parçacığı ile başlatılabilmektedir. Bu iş parçacığı bloğu da ızgara yapısını oluşturmaktadır. Birçok iş parçacığı bloğu grubu ile birden çok ızgara yapısı oluşturulabilmektedir. Bu sayede, CUDA mimarisinin mevcut kodu yürütme anında kullandığı toplam iplik sayısının en üst düzeye çıkarılmasını mümkün kılmaktadır (Bandyopadhyay, 2019).

İş parçacığı, GPU yapısındaki tek çekirdekli işleri yeniden işlemek için paralel olarak çalıştırılan birimdir. GPU iş parçacığı CPU iş parçacıklarına çok benzemektedir. Aralarındaki tek fark GPU'larda kullanılabilir iş parçacığı sayısının çok daha yüksek olabilmesidir. İş parçacığı, şekil 1 üzerinde görüldüğü gibi, bir iplik koleksiyonudur. Her bloktaki iş parçacığı sayısı GPU mimarisiyle sınırlıdır. Örneğin, Maxwell mimarisine ait bir NVIDIA GeForce Titan X GPU'da, bir bloktaki maksimum iş parçacığı sayısı 1024'tür. Ancak bu sınırlamaya rağmen, bir GPU programı kesinlikle çok sayıda bloğu kullanabilir. Bu durum, yüksek performanslı GPU'larda iş parçacığının paralel olarak yürütülmesinin en üst düzeye çıkarılmasını sağlamaktadır. Izgara NVIDIA GPU'nun hesaplama yeteneğine bağlı olarak, maksimum iş parçacığı ve blok sayısına göre blok boyutunu ayarlama imkanı sunmaktadır. Bu şekilde, GPU'nun hesaplama gücünden maksimum doluluk oranıyla en iyi şekilde faydalanılmaktadır.

GPU üzerinde yüksek performanslı bilgi işleme kodu oluşturmak ve geliştirmek için Python kullanılmaktadır (Lam ve ark., 2015) (Vaidya, 2018) (Bandyopadhyay, 2019) (Holm ve ark., 2020). Python özellikle bilgisayar bilimlerinde son yıllarda çok kullanılan ve dünyada üzerindeki araştırma toplulukları tarafından kabul görmüş bir programlama dilidir. Güçlü hesaplama yetenekleri de düşünüldüğünde GPU üzerinde kullanılan kütüphaneleri verimli bir şekilde erişim sağlayabilmektedir.

Günümüzde CUDA ve OpenCL'ye erişmenin standart yolu C ve C++ programlama dilleridir. Ancak bu dillerle kod geliştirmek zaman alıcıdır ve büyük zahmet gerektirmektedir. Python gibi daha üst düzey dillerin kullanılması GPU üzerinde geliştirme verimliliğini önemli ölçüde arttırmaktadır (Holm ve ark., 2020) (Bandyopadhyay, 2019) (Vaidya, 2018). OpenCL (Holm ve ark., 2020), Apple tarafından 2009 yılında başlatılan ücretsiz ve açık heterojen bir bilgi işleme platformudur. CUDA'nın aksine, ortak bir araç seti yoktur. OpenCL'in dezavantajı, farklı üreticilerin OpenCL uygulamaları arasında büyük farklılıklar olmasıdır (Holm ve ark., 2020). CUDA'nın bu şekilde bir dezavantajı yoktur. CUDA, GPU üzerinde kullanılan NVIDA tarafından üretilmiş C/C++ dillerini içeren OpenACC derleyicisini içermektedir. CUDA, NVIDA aygıtı sürücü üzerinde ortak araç seti ile çalıştırılmaktadır. SDK örnekler ve kütüphaneler içermektedir. Microsoft Visual Studio ve Eclipse üzerinde çalışacak bir uzantıya (Nvidia Nsight) sahiptir. Bu uzantı GPU üzerinde kod vurgulama, GPU'nun izlenmesi ve GPU bottleneck tanımlamada kullanılmaktadır (Holm ve ark., 2020) (Bandyopadhyay, 2019) (Vaidya, 2018). Bottleneck, hafızanın çalışma hızıyla işlemcinin çalışma hızı arasındaki farktan dolayı oluşan bir sorundur. İşlemcinin bilgileri işleme ve hafızandan getirme hızıyla, hafızanın bilgi gönderme hızından çok daha yüksek olması bilgi işleme hızı (GPU'nun hızından bağımsız olarak) büyük oranda düşmektedir. CUDA ve OpenCL tüm bellek alt sistemi dahil GPU donanımına tam erişim sağlayan iki programlama dilidir.

NumPy, makine öğrenimi araştırıcıları ve algoritma geliştiricileri için Python programı üzerinde bilimsel hesaplama yapmak için oluşturulmuş bir kütüphanedir. NumPy kullanılarak Python üzerinde bir makine kütüphanesi olan Scikit-learn (sklearn) kütüphanesi geliştirilmiştir (Pedregosa ve ark., 2011). Derin öğrenme hesaplamaları NumPy'ı lineer cebir hesaplamaları için kullanmaktadır. Bu yüzden NumPy uyumlu bir GPU arayüzü olan CuPy geliştirilmiştir (Okuta ve ark., 2017). CuPy'nin Python üzerinde NVIDIA GPU cihazlarını kullanmak için CUDA ve C++ kodu gerektirmeyen bir yapısı vardır. CUDA ile yüksek performanslı ve Python programı ve kütüphaneleri ile son derece uyumludur. Ayrıca CUDA' da yazılmış olan çekirdekleri de desteklemektedir.

Numba (Lam ve ark., 2015) açık kaynaklı bir JIT (Just in Time) derleyicisidir. NumPy'nin verimsiz olduğu durumları optimize etmek için geliştirilmiştir. Küçük kod değişiklikleriyle, CPU'lar ve GPU'larda Python kodunu paralelleştirmek için bir dizi seçenek sunmaktadır. Numba, Python üzerinde hesaplama yapan kullanıcılar için kodun verimli bir şekilde

parallelleştirilmesinde kullanılır. GPU'ya tam performanslı olarak erişimde kolaylık sağlamaktadır. Python üzerinde GPU'ya erişim sağlayan bir başka kütüphanede OpenCV (Kaehler ve ark., 2016)'dir. Algoritmaların GPU hızlandırmasına olanak sunan bu kütüphane, GPU mimarisinde grafik işlemlerinde kullanılmaktadır. OpenCV GPU modülü CUDA kullanılarak yazılmıştır ve NVIDIA ekosisteminden faydalanmaktadır (Vaidya, 2018). PyCuda ve PyOpenCL (Klöckner ve ark., 2012) kütüphaneleri Python programlama dili ile NVIDIA CUDA arasında bağlantı kurarlar. Bu kütüphanelerin en iyi özellikleri performanslı ve dinamik bir programlama dili ile GPU ve CPU arasında görev tamamlayıcısı olarak kullanılabilmeleridir.

3.3. Güç Tüketimi

Güç tüketimi, bilimsel hesaplama ve veri işleme ihtiyacının artması ile gelişme kaydedilmesi gerekliliği görülen konu haline gelmiştir. Bu durumlar tipik olarak gömülü ve mobil hesaplama platformlarında bulunan güç verimli mimarilerin kullanılması ve geliştirilmesi gerekliliğini ortaya koymuştur (Huynh ve ark., 2012). Elektronik devrelerde güç tüketimi sırasındaki aşırı kayıp statik ve dinamik güç kaybından kaynaklanmaktadır. Statik güç kaybı, alt eşik iletimi nedeniyle devrenin durgun halde gerçekleşmesi, boşaltma kaynağı delinmesi gibi durumlardır. Statik gücün bu kayba katkısı, dinamik güce kıyasla düşüktür (Katreepalli ve ark., 2019). Dinamik güç kaybı ise sinyal geçişi ve kısa devre akımı nedeniyle oluşmaktadır. Kısa devre güç kaybı, sinyal geçişi sırasında güç kaynağının ve toprağın anlık olarak kısalmasına bağlıdır. Kısa devre gücünün güç kaybına etkisi, dinamik gücün yaklaşık %5-10'udur. Düşük-yüksek ve yüksek-düşük sinyal geçişi nedeniyle tüketilen güç, dinamik gücün önemli bir kısmını oluşturmaktadır (Katreepalli ve ark., 2019). Denklem 1 üzerinde bir mikrodenetleyici için güç tüketim oranı Υ ile gösterilmiştir.

$$\Upsilon(\emptyset,\pi)=E/\pi \quad (1)$$

Bağlantısı ile hesaplanmaktadır. Denklem 2 üzerinde E, bir kaynağın kapanma dönemindeki güç tüketimidir (Lautner ve ark., 2018) ve denklem 2 üzerinde formülü gösterilmiştir.

$$E=(\emptyset+t_0\ )C_e \quad (2)$$

Güç verimliliği modern bilgisayarlarda en önemli ölçütlerden birisi olmuştur. Donanımın bir işlemi gerçekleştirmek için kullandığı gücü ölçmek ve mümkün olduğunca aza indirmek için kullanılan yöntemlerdir. Daha yüksek güç verimliliği ile çalışan mimariler, daha düşük hızda veya daha yüksek maliyeti olan mimariler yerine tercih edilmektedir. Bilgisayar mimarisinde güç tüketimi ölçümü MIPS / W (işlemcinin saniyede işlediği komut sayısı/ Watt) 'dir. Farklı donanımlarda enerji verimliliğini değerlendirmek bazı ölçüm alanları kullanılmaktadır Bunlardan bazıları (Qasaimeh ve ark., 2019); Çalışma zamanı: Donanımın çalışma zamanı performansı, bir kodunun başlangıcı ve bitişi arasındaki geçen süre (gecikme süresi) ölçülerek değerlendirilir.

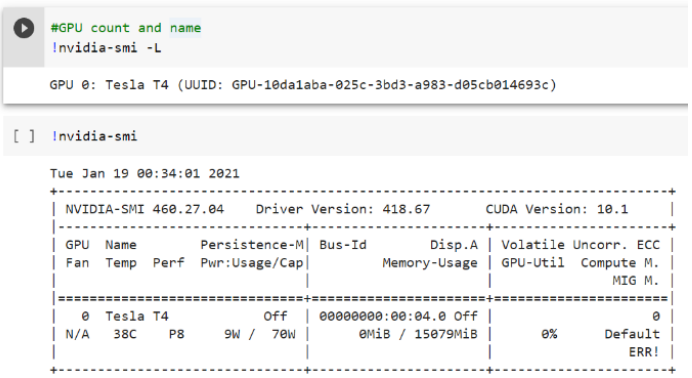
Enerji: Kod bloğu başına enerji tüketimi, bir kod bloğunun bir çekirdeğin işlemlerini gerçekleştirmek için dağıtılan elektrik enerjisi miktarını belirler. Bir çerçeveyi işlemek için gecikme süresi boyunca tüketilen güç olarak ölçülür. Cihaz gücü iki bölüme ayrılabilir:

(1) Statik güç: aktif hesaplama yapılmadığında (sistem boşta iken) tüketilen güç miktarını gösterir,

(2) Dinamik güç: sistem hesaplama yaptığı sırada statik güç seviyesinin üzerinde tüketilen güç miktarını gösterir.

Enerji Geciktirme Ürünü (EDP): Çalışma zamanı veya yalnızca kare başına enerji resmin tamamını göstermez. Bir donanım platformu pratik kullanım için çok yavaşken çok düşük güçte olabilir. Enerji Gecikme Ürünü (EDP) metriği, kare başına tüketilen enerji (mJ / kare) ile (ms / kare) olarak ölçülen algoritmanın verimini dikkate alır. EDP, enerji / çerçeve ve gecikme süresinin ürünüdür. Bu şekilde, belirli hesaplama için hangi donanım mimarisinin daha uygun olduğuna karar verirken adil bir karşılaştırma yapılabilir. Düşük EDP daha iyidir, bu da donanım mimarisinin daha kısa sürede daha az güç kullanarak belirli hesaplama görevlerini bitirebileceği anlamına gelir.

GPU'larda güç tüketimi analizinde "Sıcaklık", "Güç Tüketimi" ve "Çalıştırılma Anında Hafıza Boyutunu" parametrelerini izlemek için Python programlama dilinde Nvidia-Smi komutu kullanılmaktadır. Anlık olarak harcanan gücün ölçülmesini sağlayan Nvidia-Smi komutu şekil 2 ile verilmiştir.

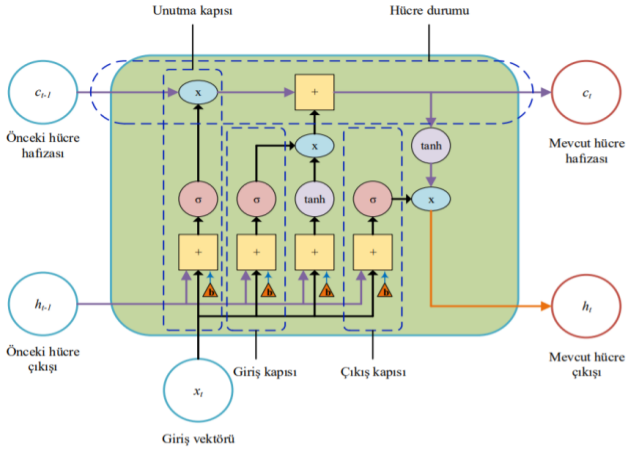


Şekil 2. Nvidia-Smi komutu kullanımı.

Nvidia-Smi GPU işlemlerinin kullanım sırasında arka planda çalışarak yapılan işlemler üzerinde GPU'nun güç tüketiminin, sıcaklığının ve hafıza boyutunun çıktısını günlük dosya şeklinde elde edilmesini sağlamaktadır. Nvidia-Smi yüksek performanslı GPU'ları desteklemektedir. Mevcut bilgisayarın günlük CPU bilgisini tutmamaktadır.

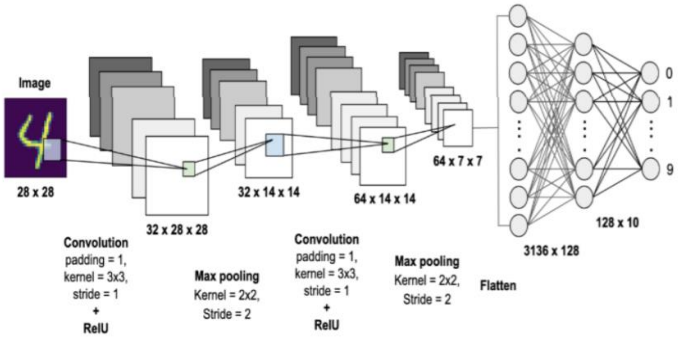
3.4. Derin Öğrenme Yöntemleri

Bu çalışma kapsamında üç farklı derin öğrenme yöntemi kullanılmıştır. Bunlardan; RNN literatürde tekrarlayan sinir ağı olarak adlandırılmaktadır. Zaman serisi ve belirli bir sıra ile alınan verileri işlemek amacıyla yaygın olarak kullanılmaktadır. İleri ve geri beslemeli işlemler ile bir döngü oluşturulur ve sıralı veriler uygulanarak başarılı sonuçlar elde edilir. RNN yönteminde büyük miktarda zaman verisine ihtiyaç duyulduğunda geri besleme işlemi sırasında sistemin türevinin aşırı azalması, yok olması veya çok yüksek değerlere ulaşması gibi sorunlar yaşanmaktadır. Bu sorunları çözmek amacıyla LSTM yöntemi geliştirilmiştir. LSTM giriş katmanı, çıkış katmanı ve RNN mimarisinde bulunmayan Unutma katmanından meydana gelmektedir. İstenmeyen veri unutma katmanı sayesinde filtrelenmekte ve büyük veri setlerinde başarılı sonuçlar elde edilmektedir. Şekil 3'te Mimarisi ve bu mimari üzerinde bulunan bölümlerin gösterimi yapılmıştır (Dursun ve ark., 2021).



Şekil 3. LSTM (Uzun Kısa Süresi Bellek) algoritmasını mimarisi.

CNN ise, temel olarak görüntüleri sınıflandırmak, benzerlikleri kullanarak kümelemek ve nesne tanıma işlemleri için kullanılan yapay sinir ağlarıdır. Çok katmanlı mimariye sahip sinir ağlarının ileri yayılım algoritması ile oluşturulmuş bir türüdür (Selvin ve ark., 2017), (Ullah ve ark., 2017), (Zhu ve ark., 2019). CNN bir veya daha fazla katmanlardan oluşmaktadır. Şekil 4 üzerinde katmanları gösterilmiştir (Bozkaya ve ark., 2021).

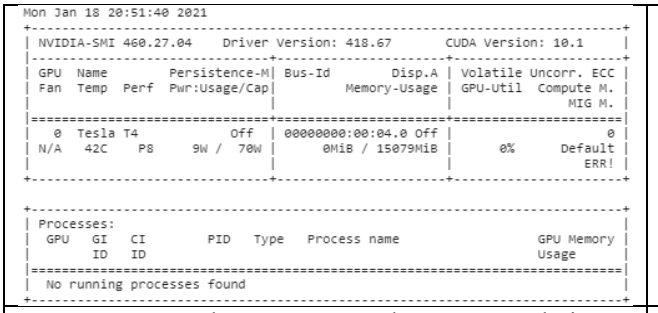


Şekil 4. CNN (Konvolüsyonel Sinir Ağı) katmanları.

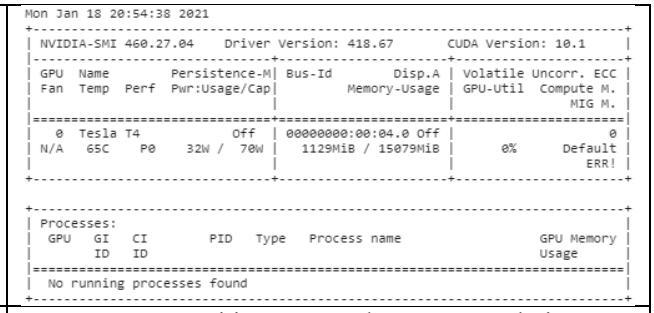
4. Deneysel Araştırma

Tablo 2. GPU üzerinde işlenen programların deneysel sonuçları.

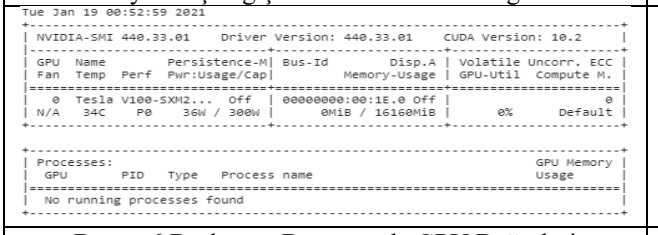
Mon Jan 18 20:51:40 2021 NVIDIA-SMI 460.27.04 Driver Version: 418.67 CUDA Version: 10.1 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. MIG M. 0 Tesla T4 Off 00000000:00:04.0 Off 0 N/A 42C P8 9W / 70W 0MiB / 15079MiB 0% Default ERR! Processes: GPU GI CI PID Type Process name GPU Memory ID ID Usage No running processes found	Mon Jan 18 20:54:38 2021 NVIDIA-SMI 460.27.04 Driver Version: 418.67 CUDA Version: 10.1 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. MIG M. 0 Tesla T4 Off 00000000:00:04.0 Off 0 N/A 65C P0 32W / 70W 1129MiB / 15079MiB 0% Default ERR! Processes: GPU GI CI PID Type Process name GPU Memory ID ID Usage No running processes found
Deney 1 Başlangıç Durumunda GPU Değerleri	Deney 1 Bitiş Durumunda GPU Değerleri
Mon Jan 18 22:06:51 2021 NVIDIA-SMI 460.27.04 Driver Version: 418.67 CUDA Version: 10.1 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. MIG M. 0 Tesla T4 Off 00000000:00:04.0 Off 0 N/A 55C P8 10W / 70W 0MiB / 15079MiB 0% Default ERR! Processes: GPU GI CI PID Type Process name GPU Memory ID ID Usage No running processes found	Mon Jan 18 22:43:39 2021 NVIDIA-SMI 460.27.04 Driver Version: 418.67 CUDA Version: 10.1 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. MIG M. 0 Tesla T4 Off 00000000:00:04.0 Off 0 N/A 73C P0 32W / 70W 2341MiB / 15079MiB 0% Default ERR! Processes: GPU GI CI PID Type Process name GPU Memory ID ID Usage No running processes found
Deney 2 Başlangıç Durumunda GPU Değerleri	Deney 2 Bitiş Durumunda GPU Değerleri
[ ] import time print(time.ctime()) Thu Jan 21 20:38:43 2021 [ ] !lscpu |grep 'Model name' # cpu modeli ve özellikleri Model name: Intel(R) Xeon(R) CPU @ 2.30GHz	[ ] print(time.ctime()) # Bitiş saati ve tarihini öğreniyoruz. Thu Jan 21 21:10:51 2021
Deney 3 Başlangıç Durumunda CPU Değerleri	Deney 3 Bitiş Durumunda CPU Değerleri
Mon Jan 18 23:05:40 2021 NVIDIA-SMI 440.33.01 Driver Version: 440.33.01 CUDA Version: 10.2 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. 0 Tesla V100-SXM2... Off 00000000:00:1E.0 Off 0 N/A 35C P0 38W / 300W 600MiB / 16160MiB 0% Default Processes: GPU PID Type Process name GPU Memory Usage	Mon Jan 18 23:15:17 2021 NVIDIA-SMI 440.33.01 Driver Version: 440.33.01 CUDA Version: 10.2 GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M. 0 Tesla V100-SXM2... Off 00000000:00:1E.0 Off 0 N/A 35C P0 38W / 300W 15333MiB / 16160MiB 0% Default Processes: GPU PID Type Process name GPU Memory Usage
Deney 4 Başlangıç Durumunda GPU Değerleri	Deney 4 Bitiş Durumunda GPU Değerleri



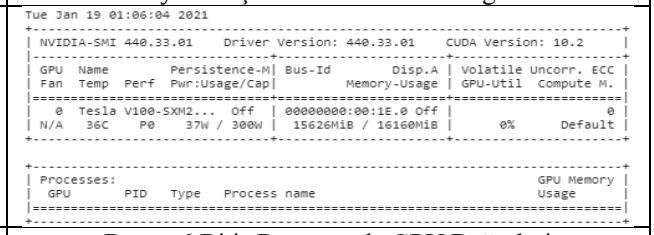
Deney 5 Başlangıç Durumunda GPU Değerleri



Deney 5 Bitiş Durumunda GPU Değerleri



Deney 6 Başlangıç Durumunda GPU Değerleri



Deney 6 Bitiş Durumunda GPU Değerleri

Araştırmada CPU ve GPU çeşitlerinin kullanılabileceği alanların ne olduğu ve bu sırada gerçekleşen işlemlerin hangi mimaride ne kadar sürede ne kadar güç harcayarak gerçekleştirildiği ölçülmüştür. Bu kapsamda nicel bir araştırma yöntemi benimsenerek çalışma yapılmıştır. Araştırma kapsamında özellikle GPU mimarisi altyapısında süre ve enerji kullanımı test edilmiş ve uygulamaların türüne göre hangi mimarinin verimli zamanlama ile tükettiği enerji üzerine vurgu yapılmıştır. Deneyler Google Colab bulut altyapısı kullanılarak sunucular üzerinde gerçekleştirilmiştir. Tablo 3 'de erişilen sunucuların listesi yer almaktadır. Bulut üzerinde başka servislerde elde dilecek güç değerleri değişiklik gösterebilmektedir.

Çalışmanın ilerleyen bölümlerinde Tablo 2 ile çalışma kapsamında gerçekleştirilen deneylerin sonuçlarından ekran görüntüleri, Tablo 3 ile uygulamalarda kullanılan mimari bilgileri ve Tablo 4 ile uygulamalarda CPU ve GPU'lar arasında performans faklılarını gözlemlemek amacıyla kullanılan algoritmalar ve bu algoritmaların sahip olduğu özellikler verilmiştir.

Tablo 2, de çalışma kapsamında gerçekleştirilen deneylerin ekran görüntüleri verilmiştir. Bu ekran görüntüleri üzerinde; deneyin başlangış ve bitiş zamanları, kullanılan donanım bilgileri, başlanıgıç ve bitiş anlarında kullanılan güç durum bilgisi ve sistemin sıcaklık bilgileri bulunmaktadır.

Tablo 2' de verilmiş olan deneylerde kullanılan mimarilerin özellikleri Tablo 3'te verilmiştir. Tablo 3 üzerinde bir adet CPU iki adet GPU bilgisi verilmiştir. CPU'nun modeli İntel Xeon 2.3GHz bir işlemcidir. GPU'lar Tesla V100-SXM2 ve Tesla T4'dür. Tesla V100-SXM2 üzerinde 18 adet CPU çekirdeği ve 5120 adet GPU çekirdeğine sahiptir. Tesla T4 üzerinde 1 adet CPU çekirdeğine sahipken 2560 adet GPU çekirdeği vardır. İntel Xeon üzerimde 1 adet CPU çekirdeği var ve GPU çekirdeği bulunmamaktadır.

Uygulamalarda CPU ve GPU mimarilerilerindeki performans faklılarını gözlemlemek amacıyla kullanılan farklı algoritmalar üzerinde deneyler yapılmıştır. Tablo 4 ile bu algoritmaların sahip olduğu özellikler verilmiştir. Bu çalışmada CNN (Evrişimsel Sinir Ağları / Convolutional Neural Networks), RNN (Tekrarlayan Yapay Sinir Ağları / Recurrent Neural Network) ve LSTM (Uzun Kısa Süreli Bellek / Long Short-Term Memory) derin öğrenme algoritmaları uygulanmıştır. Ayrıca MNIST (Tensorflow Mnist Dataset, 2021), Nvidia Time Series Dataset (Nvidia Time Series Dataset, 2021), IMDB (Tensorflow IMDB Dataset, 2021) veri setleri kullanılmıştır.

Tablo 5 ile deneyler sonucunda elde edilen ölçüm bilgileri verilmiştir. Güç kullanımları gerçekleştirilen her bir deneyin başlangıç anında ve bitiş anında ölçülmüştür.

Tablo 3. Uygulamalarda kullanılan mimariler.

	İşlemci Mimarisi	**CPU Clock**	**CPU Core**	**CUDA Core**	**GPU Memory**
İntel Xeon	İntel Xeon CPU	2.3GHz	1	-	-
Tesla V100-SXM2	İntel Xeon E5-2686 V4	2.3 GHz	18	5120	16 GB
Tesla T4	İntel Xeon CPU	2.2GHz	1	2560	16 GB

Tablo 4. Uygulamalarda kullanılan programlar.

Program No.	**Kullanılan Algoritma**	**Kullanılan Veriseti**	**Veriseti Büyüklüğü**	**Eğitim Adım Sayısı**
1	CNN	MNIST	60.000	12
2	LSTM	Nvidia Time Series Dataset	2.690	5
3	RNN	IMDB	25.000	10

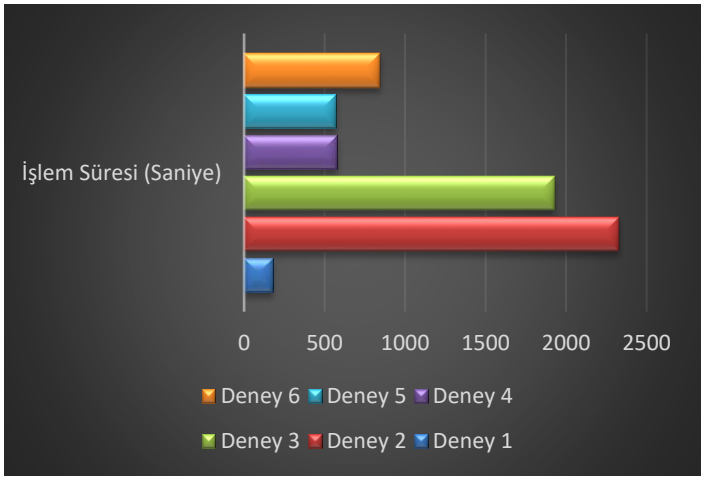
Tablo 5. Uygulama sonuçları.

Deney No.	Program No.	Mimari Adı	Toplam İşlem Süresi (saniye)	İşlem Başlangıcında Ölçülen Güç (Watt)	İşlem Bitiminde Ölçülen Güç (Watt)
1	1	GPU- Tesla T4	178	9W	32W
2	2	GPU- Tesla T4	2328	10W	32W
3	1	CPU- İntel Xeon (R) 2.3GHz	1928	3W	8W
4	2	GPU- Tesla V100-SXM2-16GB	577	38W	38W
5	3	GPU- Tesla T4	568	9W	32W
6	3	GPU- Tesla V100-SXM2-16GB	845	36W	37W

5. Sonuçlar

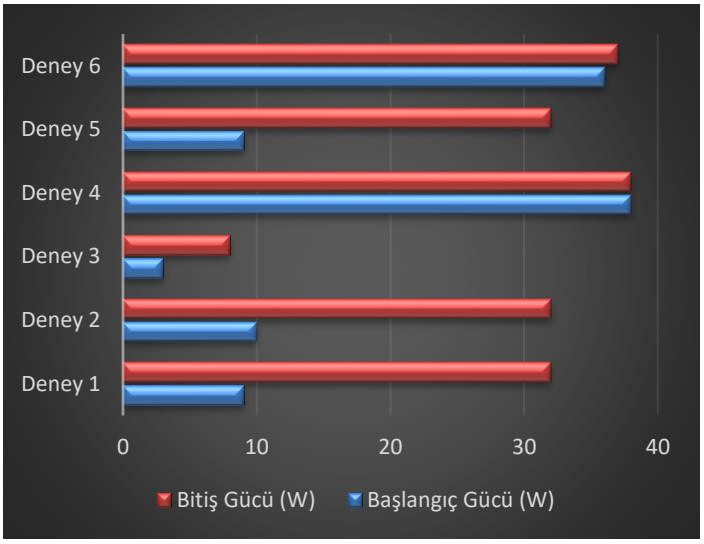
Bu çalışma kapsamında elde edilen en önemli bulgu veriseti boyutuna bağlı olarak işlemci mimarileri üzerinde güç tüketimi farklılıkları olmuştur. Alınan sonuçlara göre tablo 2 üzerinde verilen mimariler derin öğrenme uygulamalarında başarılı olmuştur. Ancak işlem süresi ve güç tüketimi bakımından farklı sonuçlar elde edilmiştir. Derin öğrenme algoritmaları ile farklı verisetleri ve mimariler üzerinde gerçekleştirilmiş uygulamalar farklı sürede tamamlanmış ve farklı seviyede güç tüketmişlerdir.

Tablo 4 üzerinde verilen deneylerin işlem süresi saniye cinsinden hesaplanmıştır. 6 adet deney yapılmıştır. Bir numaralı deneyde GPU-Tesla T4 mimarisi üzerinde Mnist veri seti kullanılarak CNN uygulaması gerçekleştirilmiştir. Bu deney 178 saniye sürmüştür. İki numaralı deney GPU-Tesla T4 mimarisinde Nvidia Time Series Dataset veri seti kullanılarak LSTM uygulaması gerçekleştirilmiştir. Bu deney 2328 saniyede tamamlanmıştır. Üç numaralı deney CPU- İntel Xeon (R) 2.3GHz mimarisi üzerinde Mnist veri seti kullanılarak CNN uygulaması gerçekleştirilmiştir. CPU üzerinde derlenen bu deney 1928 saniye sürmüştür. Dört numaralı deney GPU- Tesla V100-SXM2-16GB mimarisi üzerinde Nvidia Time Series Dataset veri seti kullanılarak LSTM uygulaması gerçekleştirilmiştir. Uygulama 577 saniyede tamamlanmıştır. Beşinci deney GPU-Tesla T4 mimarisi üzerinde derlenerek IMDB veri seti ile RNN uygulaması yapılmıştır. Bu uygulama 568 saniye sürmüştür. Altıncı deneyde GPU- Tesla V100-SXM2-16GB mimarisi üzerinde IMDB veri seti ile RNN uygulaması yapılmıştır. Bu deney 845 saniye sürmüştür. Gerçekleştirilen deneylerin işlem sürelerinin karşılaştırıldığı grafik Şekil 5 üzerinde verilmiştir.



Şekil 5. Deneylerin süre açısından karşılaştırması.

Gerçekleştirilen deneylerde her deney için ayrı ayrı başlangıç ve bitiş güç hesabı yapılmıştır. Bu hesaplama ile deneyler için harcanan güç ölçümleri ortaya çıkmıştır. Bu ölçümler; deney 1 için 23W, deney 2 için 22W, deney 3 için 5W, deney 4 için fark yok, deney 5 için 23W, deney 6 için 1W olarak ölçülmüştür. Şekil 6 üzerinde deneylerde harcanan güç miktarının karşılaştılması verilmiştir.



Şekil 6. Deneylerde harçanan güç karşılaştırıması.

Sonuç olarak, çalışma içerisinde kullanılan veri seti ve algoritmaların işlemci mimarileri üzerinde gerçekleşen güç tüketiminde etkili olduğu görülmüştür. Birinci ve üçüncü deneyde Mnist veri seti kullanılarak CNN uygulaması gerçekleştirilmesine rağmen işlem süresinde farklılık olduğu ölçülmüştür. İkinci ve dördüncü deneylerde farklı işlemci mimarileri üzerinde Nvidia Time Series veri seti kullanılarak LSTM uygulaması gerçekleştirilmiş fakat farklı işlem süresi ve güç tüketimi ölçülmüştür. Beşinci ve altıncı deneylerde farklı işlemci mimarileri üzerinde IMDB veri seti ile RNN uygulaması gerçekleştirilmiş ve farklı işlem süresi ve güç tüketimi ölçülmüştür.

Çalışmada mevcut işlemci mimarilerinde derin öğrenme uygulaması gerçekleştirildiğinde göz önünde bulundurulması gereken durumlar üzerinde deneyler yapılmıştır. Yazılım derlenirken uygulama da kullanılanılan veri seti büyüklüğü ve seçilen algoritma, mevcut işlemci mimarisi ile sınırlı kalmaktadır. Bu sınırın özellikle GPU mimarilerinde güç tüketimi ve çalışma süresini, CPU mimarisinde çalışma süresi bakımından etkilediği görülmüştür. Bu yüzden derin öğrenme

uygulamalarında etkili güç tüketimi için veri setinin büyüklüğü göz önünde bulundurularak uygun mimari seçilmelidir.

Kaynakça

Själander, M., Martonosi, M., & Kaxiras, S. (2014). Power-efficient computer architectures: Recent advances. Synthesis Lectures on Computer Architecture, 9(3), 1-96.

Huang, L., Lü, Y., Ma, S., Xiao, N., & Wang, Z. (2019). SIMD stealing: Architectural support for efficient data parallel execution on multicores. Microprocessors and Microsystems, 65, 136-147.

Li, T., Evans, A. T., Chiravuri, S., Gianchandani, R. Y., & Gianchandani, Y. B. (2012). Compact, power-efficient architectures using microvalves and microsensors, for intrathecal, insulin, and other drug delivery systems. Advanced drug delivery reviews, 64(14), 1639-1649.

Katreepalli, R., & Haniotakis, T. (2019). Power efficient synchronous counter design. Computers & Electrical Engineering, 75, 288-300.

Dehnavi, M., & Eshghi, M. (2018). Cost and power efficient FPGA based stereo vision system using directional graph transform. Journal of Visual Communication and Image Representation, 56, 106-115.

Huynh, T. V., Mücke, M., & Gansterer, W. N. (2012). Evaluation of the Stretch S6 Hybrid Reconfigurable Embedded CPU Architecture for Power-Efficient Scientific Computing. Procedia Computer Science, 9, 196-205.

Lautner, D., Hua, X., DeBates, S., Song, M., & Ren, S. (2018). Power efficient scheduling algorithms for real-time tasks on multi-mode microcontrollers. Procedia computer science, 130, 557-566.

Wyant, C. M., Cullinan, C. R., & Frattesi, T. R. (2012). Computing performance benchmarks among cpu, gpu, and fpga. Computing.

Mittal, S., & Vetter, J. S. (2014). A survey of methods for analyzing and improving GPU energy efficiency. ACM Computing Surveys (CSUR), 47(2), 1-23.

Betkaoui, B., Thomas, D. B., & Luk, W. (2010). Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing. In 2010 International Conference on Field-Programmable Technology (pp. 94-101). IEEE.

Mumcu, M. C., & Bayar, S. (2020). Parallel Implenetation Of The GPR Techniques For Detecting And Mapping Ancient Buildings By Using CUDA. Avrupa Bilim ve Teknoloji Dergisi, 352-359.

Stratton, J. A., Anssari, N., Rodrigues, C., Sung, I. J., Obeid, N., Chang, L., ... & Hwu, W. M. (2012). Optimization and architecture effects on GPU computing workload performance. In 2012 Innovative Parallel Computing (InPar) (pp. 1-10). IEEE.

InAccel. (2018). Cpu Gpu Fpga or Tpu, https://medium.com/@inaccel/cpu-gpu-fpga-or-tpu-which-one-to-choose-for-my-machine-learning-training-948902f058e0, 15.03.2021.

Kahoul, A., Constantinides, G. A., Smith, A. M., & Cheung, P. Y. (2009). Heterogeneous architecture exploration: Analysis vs. parameter sweep. In International Workshop on Applied Reconfigurable Computing (pp. 133-144). Springer, Berlin, Heidelberg.

Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., & Jones, P. H. (2019). Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. In 2019 IEEE International Conference on Embedded Software and Systems (ICESS) (pp. 1-8). IEEE.

Holm, H. H., Brodtkorb, A. R., & Sætra, M. L. (2020). GPU computing with Python: Performance, energy efficiency and usability. Computation, 8(1), 4.

Bandyopadhyay, A., (2019). Hands-On GPU Computing with Python: Explore the capabilities of GPUs for solving high performance computational problems, Packt Publishing, ISBN-13: 978-1789341072

Aydın, S., Samet, R., & Bay, Ö. F. (2020) Gpu Programlamada Cuda Platformu Kullanılan Paralel Görüntü İşleme Çalışmalarının İncelenmesi. Politeknik Dergisi, 23(3), 737-754.

Vaidya, B. (2018). Hands-On GPU-Accelerated Computer Vision with OpenCV and CUDA: Effective techniques for processing complex image data in real time using GPUs. Packt Publishing Ltd.

Goz, D., Ieronymakis, G., Papaefstathiou, V., Dimou, N., Bertocco, S., Simula, F., ... & Taffoni, G. (2020). Performance and energy footprint assessment of FPGAs and GPUs on HPC systems using Astrophysics application. Computation, 8(2), 34.

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (pp. 1-6).

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). Cupy: A numpy-compatible library for nvidia gpu calculations. In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS) (p. 7).

Kaehler, A., & Bradski, G. (2016). Learning OpenCV 3: computer vision in C++ with the OpenCV library. " O'Reilly Media, Inc.".

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., & Fasih, A. (2012). PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Computing, 38(3), 157-174.

Sriadibhatla, S., & Baboji, K. (2019). Design and implementation of area and power efficient reconfigurable fir filter with low complexity coefficients. Gazi University Journal of Science, 32(2), 494-507.

Öztürk, A., Allahverdı, N. & Saday, F. (2022). Application of artificial intelligence methods for bovine gender prediction. Turkish Journal of Engineering, 6 (1), 54-62.

Karadağ, N., Çetinkaya, A. & Aydın, H. (2020). Yerel İkili Desenler Histogramları ile Covid-19 Tanılı Kişiler Üzerinde Kimlik Analizi ve Bildiri Sistemi. Gazi Mühendislik Bilimleri Dergisi, 6 (3), 172-183.

Yılmaz, O., Aydın, H. & Çetinkaya, A. (2020). Faster R-CNN Evrişimsel Sinir Ağı Üzerinde Geliştirilen Modelin Derin Öğrenme Yöntemleri ile Doğruluk Tahmini ve Analizi: Nesne Tespiti Uygulaması. Avrupa Bilim ve Teknoloji Dergisi, (20), 783-795.

Nabiyev, V. V. (2003). Yapay zeka: problemler-yöntemler-algoritmalar. Seçkin Yayıncılık.

Balci, M., Tasdemir, S., Ozmen, G., & Golcuk, A. (2022). Machine Learning-Based Detection of Sleep-Disordered Breathing Type Using Time and Time-Frequency Features. Biomedical Signal Processing and Control, 73, 103402.

Abas, A. İ., Kocer, H. E., & Akhan Baykan, N. (2021). Medical image fusion with convolutional neural network in multiscale transform domain. Turkish Journal of Electrical Engineering & Computer Sciences, 29.

Uysal, E., Yumusak, S., Oztoprak, K., & Dogdu, E. (2017). Sentiment analysis for the social media: A case study for turkish general elections. In Proceedings of the SouthEast Conference (pp. 215-218).

Örnek, M. N., & Örnek, H. K. (2021). Developing a deep neural network model for predicting carrots volume. Journal of Food Measurement and Characterization, 1-9.

Koklu, M., Unlersen, M. F., Ozkan, I. A., Aslan, M. F., & Sabanci, K. (2021). A CNN-SVM study based on selected deep features for grapevine leaves classification. Measurement, 110425.

Lee, Y., & Shim, J. (2019). Deep learning and color histogram based fire and smoke detection research. International journal of advanced smart convergence, 8(2), 116-125.

Bozkaya, F., Yusefı, A., Tığlıoğlu, Ş., Kaya, A. K., Kazancı, O., Akmaz, M. Y., Durdu, A. & Sungur, C. (2021). Otonom Sistemlerde Veri Çoğaltma Yöntemleri Kullanılarak İyileştirilmiş Gerçek Zamanlı Nesne Tespiti. Avrupa Bilim ve Teknoloji Dergisi, Ejosat Special Issue 2021 (ICCEES), 83-87.

Dursun, Ö. O. & Toraman, S. (2021). Uzun Kısa Vadeli Bellek Yöntemi ile Havayolu Yolcu Tahmini. Journal of Aviation, 5 (2), 241-248.

Sanapala, K. (2017). Two Novel Subthreshold Logic Families for Area and Ultra Low-Energy Efficient Applications: DTGDI & SBBGDI. Gazi University Journal of Science, 30(4), 283-294.

Demirbas, A. A., & Çınar, A. (2020) Nesne Sınıflandırma İşlemi İçin Tensor İşleme Birimi ve Cpu Performans Karşılaştırması. Bilgisayar Bilimleri ve Teknolojileri Dergisi, 1(1), 10-15.

Dodiu, E., & Gaitan, V. G. (2012). Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers—Concept and theory of operation. In 2012 IEEE International Conference on Electro/Information Technology (pp. 1-5). IEEE.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate?. In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (pp. 256-267).

Çetin, N. M., & Hacıömeroğlu, M. (2013). Gpu Hızlandırmalı Veri Demetleme Algoritmalarının İncelenmesi. Ajıt-E: Bilişim Teknolojileri Online Dergisi, 4(11), 19-59.

Nvidia Time Series Dataset, Modeling Time Series Data with Recurrent Neural Networks in Keras, Son Erişim Tarihi: 02.01.2021, https://courses.nvidia.com/courses/course-v1:DLI+L-FX-24+V1/about adresinden erişildi.

Tensorflow Mnist Dataset, Loads the MNIST dataset, Son Erişim Tarihi: 05.01.2021, https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data adresinden erişildi.

Tensorflow IMDB Dataset, Loads the IMDB dataset, Son Erişim Tarihi: 09.01.2021, https://www.tensorflow.org/api_docs/python/tf/keras/datasets/imdb/load_data adresinden erişildi.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In 2017 international conference on advances in computing, communications and informatics (icacci) (pp. 1643-1647). IEEE.

Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action recognition in video sequences using deep bi-directional LSTM with CNN features. IEEE access, 6, 1155-1166.

Zhu, F., Ye, F., Fu, Y., Liu, Q., & Shen, B. (2019). Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. Scientific reports, 9(1), 1-11.