ISTANBUL MEDENIYET UNIVERSITY
INSTITUE OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Indoor Surface Type-based Task Assignment in Heterogeneous Multi-Robot Systems

Master Thesis

Asiye Demirtaş

July 2023

ISTANBUL MEDENIYET UNIVERSITY
INSTITUE OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Indoor Surface Type-based Task Assignment in Heterogeneous Multi-Robot Systems

Master Thesis

Asiye Demirtaş

Supervisor

Asst. Prof. Haluk Bayram

Co-supervisor

Assoc. Prof. Gökhan Erdemir

July 2023

THESIS JURY APPROVAL

This Master thesis titled "Indoor Surface Type-based Task Assignment in Heterogeneous Multi-Robot Systems" written by Asiye Demirtaş at the Department of Electrical and Electronic Engineering was accepted by our jury.

JURY MEMBERS	**SIGNATURE**
Supervisor:	
Asst. Prof. Haluk Bayram Istanbul Medeniyet University	.....
Other Jury Members:	
Asst. Prof. İbrahim Genç Istanbul Medeniyet University	.....
Prof. Dr. Ahmet Emin Kuzucuoğlu Marmara University	.....

Date of Defense: 13/07/2023

STATEMENTS

Style and Reference Manual Statement

Having reviewed this thesis written under my supervision, I confirm that it has been written in accordance with APA Manual of Style and used its [footnote/in text] reference format consistently throughout the entire text.

Asst. Prof. Haluk Bayram

Declaration of Originality

I hereby declare that all information in this dissertation has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conducts, I have fully cited and referenced all material and results that are not original to this work.

Asiye Demirtaş

ACKNOWLEDGEMENTS

I want to express my gratitude to Asst. Prof. Haluk Bayram for his guidance and support throughout the thesis period, and I would also like to thank my advisor for expanding my horizons since my undergraduate years.

I also would like to thank Assoc. Prof. Gökhan Erdemir for his guidance, encouragement, and sincere behavior by adopting me as a part of his family.

I would like to thank Asst. Prof. İbrahim Genç, whose lectures during my undergraduate studies broadened my perspectives, and also I express my thanks because he attended as a jury member in my thesis defense.

I would like to thank Prof. Ahmet Emin Kuzucuoğlu, who was on the jury in my thesis defense, for his support.

I would like to thank esteemed Prof. İsmail Küçük, who always behaves in a friendly manner since the day I started working at Istanbul Sabahattin Zaim University (IZU), for his support and sincerity.

I would like to thank esteemed Prof. Nizamettin Erduran and Asst. Prof. Oktay Doğangün at IZU NarLab for their sincerity, support in technical manners, and good conversations that comforted me during this intense thesis period.

I would like to thank my dear friend and colleague Erdal Alimovski for his dedication, encouragement, and sincerity, and I also appreciate his day-and-night support during my thesis period.

I would like to thank my beloved friend Yağmur Uşak, who has always given me strength and not made me feel alone, both during the thesis period and in my private life, for her support throughout this process.

I would like to thank my precious family, who loves, cares, and supports me at all

times and under all circumstances, and my precious niece Zümra Sena, who brings joy to Narlab.

This work was supported by Scientific Research Projects (BAP) through the Istanbul Sabahattin Zaim University, Istanbul, Türkiye, under BAP-1000-88.

GENİŞ ÖZET

Heterojen Çoklu Robot Sistemlerinde İç Mekan Yüzey Tipi Tabanlı Görev Atama

Asiye DEMİRTAŞ

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Haluk BAYRAM

Eş Danışman: Doç. Dr. Gökhan ERDEMİR

Temmuz, 2023. 58 sayfa.

Teknolojideki hızlı gelişmelerle birlikte iç mekanlarda mobil robotların kullanımı yaygınlaşmaktadır. Ev, hastane ve depo gibi çeşitli iç mekan ortamlarında iş yapılma şeklini değiştirerek robotların nakliye, teslimat ve temizlik gibi çeşitli faaliyetleri gerçekleştirmesi mümkün kılınmıştır. İç mekan mobil robotların çalışma ortamlarındaki nesneler ve yüzeylerle etkileşime girebilmeleri en önemli özelliklerinden biridir. Özellikle yüzey sınıflandırma işlemi robotun hareketini, kararlılığını ve güvenliğini sürdürmek için çok önemlidir. Kapalı alanlarda yüzeylerin karakteristikleri yoğun değişkenlik gösterir. Halı, fayans, ahşap vb. farklı yüzey türleri iç mekan mobil robotlarının hareketi için farklı zorluklar yaratmaktadır. Kapalı yüzeyler tipik olarak düzdür ve hareket etmesi kolaydır. Ancak bazı alanlar kaygan ve engebeli olabilir, bu da robotun hareketini zorlaştırabilir ve istenmeyen durumların meydana gelmesine neden olabilir. Robotlar, yüzey türlerini doğru bir şekilde sınıflandırırken aynı zamanda yüzeylerde nasıl hareket edeceklerini akıllıca seçebilir ve buna göre hareketlilik parametrelerini değiştirebilirler.

Bu tezin temel amacı, en doğru yüzey sınıflandırma modelini belirlemek ve yüzey tipine göre robotlarda görev paylaşımı yapmaktır. Bildiğimiz kadarıyla çeşitli yüzey tiplerini içeren açık kaynaklı iç mekan yüzey veri seti yoktur. Araştırmalarda kullanılan veri setlerinin çoğu birkaç dış yüzey tipini ve bir veya en fazla iki iç yüzey tipini içerir. Bu nedenle benzersiz bir iç mekan yüzey veri seti oluşturulup açık kaynak haline getirilmesi amaçlanmaktadır. Oluşturulan veri seti halı, fayans ve ahşap gibi üç yüzey tipinden oluşmaktadır. Yüzey örnekleri çeşitli iç ortamlarda toplanmıştır. Bu nedenle her yüzey farklı renk ve tiplerden oluşmaktadır. Veri seti 870 halı, 638 fayans ve 573 ahşap olmak üzere toplam 2081 görüntü içermektedir.

Daha sonra veri setimiz yüzey sınıflandırma görevini gerçekleştirmek için InceptionV3,

VGG16, VGG19, ResNet50, Xception, InceptionResNetV2, MobileNetV2 ve önerilen MobilenetV2-Modified gibi son teknoloji CNN modelleri üzerinde eğitilmiştir. Her model mimarisi, katman sayısı, özellik çıkarma ve öğrenme için özel işlemler açısından farklılık göstermektedir. Örneğin, InceptionV3 model mimarisi 47 katmandan oluşur ve 3x3'lük konvolüsyonları 3x1 ve 1x3'e ayırır. Böylece bu işlem, belirli detayların daha etkin bir şekilde öznitelik çıkarımına ve daha kısa işlem süresine olanak tanır. VGG16 ve VGG19 modelleri 1 adımlı 3x3 filtreli evrişimli katmanlara ve 2x2 filtreli maksimum havuzlama katmanına odaklanmaktadır. İsimlerinden de anlaşılacağı gibi, VGG19'un mimarisi 19 katman, VGG16 ise toplam 16 katmandan oluşmaktadır. ResNet50'nin ana mimarisi, VGG mimarisinden esinlenmiştir, ancak VGG'den farklı olarak bağlantıları atlama (skip connections) adı verilen bir teknik içermektedir. ResNet50 mimarisi, her biri üç evrişim katmanına sahip bir kimlik bloğuna (identity block) ve üç evrişim katmanına sahip bir evrişim bloğuna olmak üzere beş aşama içermektedir. Xception modelinin geliştirilmesinde Inception modelinden esinlenilmiştir. Xception model mimarisini, yan yana işlev gören derin ve kapsamlı evrişimli katmanlar oluşturur. Mimarinin derinliği 71 katmandır. InceptionResNetV2, Inception ve artık ağların (residual networks) bir birleşimidir. InceptionResNet modülünün mimarisi iki ana bölüme ayrılmıştır: kısayol bağlantıları (shortcut connections) ve artık bloklar (residual blocks). Sınırlı hesaplama gücüne sahip mobil ve gömülü cihazlarda CNN kullanılarak çeşitli görevleri gerçekleştirmek için MobileNetV2 mimarisi önerilmiştir. MobileNetV2 modelinin ana mimarisi, darboğaz seviyeleri (bottleneck levels), ters artık yapılar (inverse residual structure) ve bağlantılar (connections) gibi üç temel bölümden oluşmaktadır. Bu metodoloji sayesinde sınırlı belleğe sahip cihazlarda gerçek zamanlı sınıflandırma mümkün olmuştur. Bu tez çalışmasında hafif bir mimariye sahip MobileNetV2-Modified modeli öneriyoruz. MobileNetV2 orijinal modelinin ana mimarisinde mimarinin son 11 katmanı kaldırılmıştır. Kaldırılan katmanların yerine 16 filtreli ve 5x5 çekirdek boyutuna sahip 2B-evrişimsel katmanı, bırakma (dropout katmanı), havuzlama (global average pooling) katmanı, ve iki tam bağlı katmanları eklenmiştir. Böylece ana mimaride değişiklikler yapılarak toplam blok sayısı 17'den 16'ya düşürülmüştür. Uygulanan son teknoloji CNN modellerinin her biri önerilen model dışında önceden eğitilmiş modeller olarak kullanılmıştır. Önerilen modelde son 23 katmanı eğitilebilir hale getirilmiştir. Önerilen model, diğer CNN modellerine kıyasla daha kısa eğitim süresi ve daha az eğitim parametresi sunmaktadır.

Yüzey sınıflandırma görevi için en etkili ağı belirlemek amacıyla SGD, RMSProp, Adam, Adadelta, Adamax, Adagrad ve Nadam gibi birkaç optimizasyon algoritmasının etkisini araştırarak analizi genişlettik. Yani her bir CNN modelini 7 farklı optimizasyon algoritması ile eğittik. Derin öğrenmede en çok kullanılan optimize edicilerden biri SGD'dir. SGD, tüm eğitim setini güncellemek yerine eğitim verilerinin alt kümesindeki parametreleri günceller. Adam, Adagrad ve RMSProp'un birleşimidir. Hem birinci anı hem de ikinci anı birleştiren Adam, öğrenme oranını her parametrenin özel gereksinimlerine uyarlayarak optimizasyon performansını geliştirir. Hesaplama açısından oldukça verimlidir ve çok az bellek gerektirir. RMSProp, sağlamlığı ve performansı nedeniyle derin öğrenmede popülerlik kazanmıştır. RMSProp, kare gradyanların çalışan ortalamasını koruyarak her parametre için uyarlanabilir bir öğrenme oranı hesaplar. Bunu yaparak RMSProp her parametre için öğrenme oranını ayrı ayrı ayarlayarak modelin çeşitli bölümlerinde etkili güncellemeler sağlar. Adagrad, eğitim sürecinde çeşitli parametreler için uygun bir öğrenme oranı seçmenin zorluğunu gidermek için önerilmiştir. Adagrad'ın temel amacı, öğrenme oranını her parametre için kare gradyanların toplamının kareköküyle ters orantılı olarak ayarlamaktır. Adagrad bu özellikleri nedeniyle az verilerle ve farklı öneme sahip özelliklerle çalışmada etkilidir. Adadelta, yalnızca gradyanların birikmiş geçmiş bilgisini kullanarak, küresel bir öğrenme hızına olan ihtiyacı ortadan kaldırmaya çalışmaktadır. Algoritma geçmiş kareli gradyanların azalan ortalaması ve güncellemelerin üssel olarak azalan ortalaması adı altında iki ana bileşen kullanmaktadır. Böylece, bu bileşenler tarafından parametre güncellemeleri hesaplanır. Özellikle bellek etkinliğinin önemli olduğu durumlarda Adadelta güçlü performans gösterir. Adamax, Adam optimizasyon algoritmasının bir çeşididir. Özellikle belleğin sınırlı olduğu durumlarda, Adam'a göre daha güvenilir ve hafıza açısından verimli bir alternatif sunmak amaçlanmaktadır. Adam ile karşılaştırıldığında Adamax, ikinci an tahmini sağlaması gerekmediğinden daha az bellek gerektirir. Nadam, Adam ve Nesterov Accelerated Gradient (NAG) kavramlarını birleştirir. Nesterov momentum tekniğinin kullanımıyla Nadam, uyarlanabilir öğrenme hızları ile Adam'ın momentumunu birleştirmektedir. Nadam, özellikle karmaşık yapılara sahip derin öğrenme modellerinde uyarlanabilmektedir. Büyük ölçekli veri kümelerine karşı öğrenme hızları, momentum ve Nesterov ivmesinin avantajlarını entegre ederek daha hızlı yakınsama ve iyileştirilmiş optimizasyon performansı sağlamaktadır. CNN modelleri yukarıda bahsedilen her bir optimizasyon algoritması ile eğitildi ve bu sayede en iyi modeli be-

lirlemiş olduk.

Bu tezde mobil robotlardan Kobuki ve Burger kullanılmıştır. Kobuki ve Burger robotları, farklı yönlerde hareket etmelerini ve gezinmelerini sağlayan bir diferansiyel tahrik sistemine sahiptir. Her iki robot da uzun pil ömrü, güvenilir odometri sensörleri ve aktüatörler gibi özellikler sayesinde çeşitli iç mekan görevleri için pratiktir. Kobuki robot ağırlığı 3 kg civarında, 35 cm çapında ve maksimum 0.7 m/s hızında hareket edebiliyor. Burger robotunun maksimum hızı 0.22 m/s'dir. Burger robotunun ağırlığı ise sensörler, pil, RaspberryPi dahil olmak üzere 1 kg civarındadır. Burger robotu 138 mm uzunluk, 178 mm genişlik ve 192 mm yüksekliktedir. Görevi yüzey tipine göre robotlar arasında paylaşmak için her bir robotun halı, fayans ve ahşap zemin üzerindeki performansını inceledik. İlk olarak, robotların performansını doğru bir şekilde analiz etmek için her yüzeyde başlangıç ve bitiş noktası olan parkur tanımladık. Parkur toplam uzunluğu 5 metre olarak ayarlandı. Parkur hazır olduğunda Kobuki ve Burger robotlarını hareket ettiriyoruz. Robotları çeşitli hızlarda test ettik. Düşük hareket hızı nedeniyle Burger 0.1, 0.15 ve 0.2 m/s hızlarda test edilirken Kobuki 0.1, 0.2 ve 0.3 m/s hızlarda test edildi. Robotların performans analizleri açısal sapma, hedef noktaya kadar olan hata oranı (cm) ve zaman açısından yapılmıştır. Açısal sapma, robotların üzerine monte edilen WT901 IMU sensörü ile ölçülmüştür. Doğru yönlendirme, hızlanma ve açısal hız ölçüm yetenekleri ile WT901, çeşitli uygulamalar için hareket algılama yetenekleri sunar. Hata oranını varış noktasına doğru ve hızlı bir şekilde ölçmek için elde taşınan bir lazer metre cihazı kullanıldı. Robotların başlangıç noktasından bitiş noktasına kadar olan hareket süreleri yazılım aracılığıyla ölçülmüştür. Veriler IMU, lazer ölçer ve yazılımdan toplandıktan sonra, hangi robotun hangi yüzeyde daha iyi performans gösterdiğini belirlemek için Analitik Hiyerarşi Süreci (AHP) karar verme yöntemi uygulanmıştır. AHP, karmaşık seçim konularını metodik olarak analiz etmek, kriterlerin ve alternatiflerin göreceli ağırlığını hesaba katmak için resmi bir çerçeve sunar. Bu nedenle AHP uygulanarak puanlama yapılmış ve hangi robotun hangi yüzeyde daha iyi olduğu tespit edilmiştir.

Bu tezdeki deneyler birkaç aşamadan oluşmaktadır. Oluşturulan veri seti üzerinde ilk olarak yüzey sınıflandırma deneyleri yapılmıştır. İkinci olarak, uygulanan son teknoloji CNN modelleri ve önerilen model arasından yüzey sınıflandırmasında en iyi perfor-

mansa sahip model gerçek zamanlı test edilmiştir. Üçüncü olarak hangi robotun hangi yüzeyde daha iyi performans gösterdiğine karar vermek için robotların performans analizi yapıldı. Son olarak robotların yüzey tipine göre görev dağılımı ile ilgili deneyler yapılmıştır.

Eğitim ve test anlamında yüzey sınıflandırma deneyleri, AMD Ryzen 9 işlemci ve Nvidia Ge Force RTX 3080 Ti ekran kartına sahip bilgisayarda gerçekleştirilmiştir. Uygulanan her bir CNN modeli ve önerilen model, bu tezde oluşturulan ve toplam 2.081 görüntü içeren veri seti üzerinde eğitilmiş ve test edilmiştir. Veri setinin sırasıyla %80-10-10 oranında eğitim, doğrulama ve test setlerine ayrılmıştır. Her modelin verimliliği ve performansı doğruluk, kesinlik, hatırlama ve F1 skoru metrikleri ile ölçülür. Ek olarak, modellerin performansını ayrıntılı olarak analiz etmek için her modelin karışıklık matrislerini gösterdik. Eğitim ve test deneysel sonuçları, önerilen modelin bu tezde uygulanan diğer tüm modellerden ve literatürdeki mevcut yaklaşımlardan daha iyi performans gösterdiğini saptanmıştır. Önerilen model, Adamax optimizer ile eğitildiğinde %99,94 eğitim doğruluğu ve %99,52 test doğruluğu elde etti. Kesinlik, hatırlama ve F1 puanı sırasıyla %99, %100 ve %100 olmuştur. Önerilen modelin eğitim kaybının her bir model arasında en düşük olan %0,53 olduğunu belirtmekte fayda var. Karışılık matrisi incelendiğinde, önerilen model bir görüntü dışında sınıflardaki her görüntüyü doğru tahmin etmiştir. InceptionV3 %95,67 test doğruluğu ile önerilen modele en yakın performansı göstermiştir. ResNet50 modeli ise %58,17 test doğruluğu ile en kötü performans sergilediği görülmüştür. Ayrıca modellerin eğitim sürelerini ve ağırlık boyutları da ölçülmüştür. Elde edilen sonuçlar, önerilen model ağırlık boyutunun orijinal MobileNetV2 modelinden 4 kat daha az olacak şekilde 11MB'a düşürülmüştür. Önerilen modelin eğitim süresi 1:25 dakika, MobileNetv2 modelinin eğitim süresi ise 1:43 dakika olarak gerçekleşmiştir.

Önerilen modelin gerçek zamanlı sınıflandırma görevinde mobil robot üzerinde çeşitli iç ortamlarda değerlendirilmiştir. Ofis, oda, koridor gibi 11 farklı ortamda yüzey sınıflandırması yapılmıştır. Gerçek zamanlı sınıflandırma yapmak için ağırlıkları dizüstü bilgisayara aktardık, ardından bilgisayar robotun üstüne yerleştirilmiştir. Test aşaması için 11 senaryo belirledik ve böylece Kobuki robotu ahşap-halı-fayans, halı-fayans ve halı-ahşap gibi yüzeylerin bulunduğu parkurlarda hareket ettirdik. Mobil robot

hareket halindeyken her 5 saniyede bir Logitech kameradan görüntü alındı ve halı, fayans veya ahşap olarak sınıflandırıldı. Toplamda 267 fotoğraf çekildi ve test edildi. 265 adet yüzey görüntüsü önerilen model tarafından doğru olarak tanınırken sadece 2 adet görüntü hatalı olarak tanınmıştır. Önerilen model, gerçek zamanlı sınıflandırmada %99,25 doğruluk elde etmiştir. Gerçek zamanlı ve çeşitli ortamlarda yüksek doğruluk elde edilmesi, önerilen modelin sağlamlığını göstermektedir. Kobuki ve Burger robotları için IMU açı sapma verileri, hedefe olan hata değeri ve başlangıç noktasından bitiş noktasına kadar geçen süre her iki robotun da üç farklı zemin üzerinde hareket etmesinden sonra elde edilmiştir.

Herhangi bir yöntem uygulamadan sadece değerlerin analizi ile hangi robotun hangi yüzeyde daha iyi performans gösterdiğine karar verilebilir. Ancak daha doğru karar verebilmek için elde edilen değerlere AHP yöntemi uygulanmıştır. Yapılan deneylerde 0,2 m/s hız için Kobuki robotun performansı halı ve fayans yüzeylerde, Burger ise ahşap yüzeyde daha iyi olmuştur. Öte yandan, 0,1 m/s hızla Burger'in performansı Fayans ve Ahşap üzerinde daha iyiyken, Kobuki halı üzerinde daha etkili olmuştur.

Ayrıca, robotların yüzey tipine göre hareket etme deneyleri gerçekleştirilmiştir. Bu hareket deneyleri için dört farklı senaryo oluşturulmuştur. Bu senaryolar sırasıyla: fayans-ahşap, halı-ahşap, sadece ahşap ve sadece halı olmak üzere belirlenmiştir. Senaryolar esnasında robotlar birbirleri ile haberleşerek yüzey türüne göre hareket etmiştir. Gerçekleşen deneylerde, her iki robot da başarılı performans sergilediği görülmüştür.

Anahtar Kelimeler: yüzey sınıflandırma, mobil robotlar, evrişimsel sinir ağları, görev atama, iç mekan robotiği, haberleşme.

ABSTRACT

Indoor Surface Type-based Task Assignment in Heterogeneous Multi-Robot Systems

Asiye DEMİRTAŞ

Master Thesis, Electrical and Electronic Engineering Department

Supervisor: Asst. Prof. Haluk Bayram

Co-supervisor: Assoc. Prof. Gökhan Erdemir

July, 2023. 58 pages.

The ability to recognize the surface type is crucial for both indoor and outdoor mobile robots. Knowing the surface type can help indoor mobile robots move more safely and adjust their movement accordingly. However, recognizing surface characteristics is challenging since similar planes can appear substantially different; for instance, carpets come in various types and colors. To address this inherent uncertainty in vision-based surface classification, this study first generates a new, unique data set composed of 2081 surface images (carpet, tiles, and wood) captured in different indoor environments. Secondly, the pre-trained state-of-the-art deep learning models, namely InceptionV3, VGG16, VGG19, ResNet50, Xception, InceptionResnetV2, and MobileNetV2, were utilized to recognize the surface type. Additionally, a lightweight MobileNetV2-modified model was proposed for surface classification. The proposed model has approximately four times fewer total parameters than the original MobileNetV2 model, reducing the size of the trained model weights from 42 MB to 11 MB. Thus, the proposed model can be used in robotic systems with limited computational capacity and embedded systems. Finally, several optimizers, such as SGD, RMSProp, Adam, Adadelta, Adamax, Adagrad, and Nadam, are applied to distinguish the most efficient network. Experimental results demonstrate that the proposed model outperforms all other applied methods and existing approaches in the literature by achieving 99.52% accuracy and an average score of 99.66% in precision, recall, and F1-score. Besides this, the proposed lightweight model was tested in real-time on a mobile robot in 11 scenarios consisting of various indoor environments such as offices, hallways, and homes, resulting in an accuracy of 99.25%. The performance analysis of the Kobuki and Burger robots on different surfaces in terms of angular deviation, time of reaching to the desired point, and sensitivity of reaching the desired point was conducted. The angular deviation was measured by using Inertial Measurement Unit (IMU) sensor. In order to assign tasks to

the robots according to the ground type, we evaluated the performances of the Kobuki and Burger robots by using Multi Criteria Decision Making Method. According to the performance analysis, it is concluded that the Kobuki robot has better performance on carpet and tile, while the Burger performs better on wood. In addition, task assignment experiments were carried out with multi robots according to the surface type. During the experiments, it was observed that both robots completed the task correctly and in accordance with the performance analysis method.

Keywords: surface classification, mobile robots, convolutional neural network, task assignment, indoor robotics, communication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ADAM	Adaptive Moment Estimation
AHP	Analytic Hierarchy Process
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Process Unit
DL	Deep Learning
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
LSTM	Long Short-Term Memory
MCDM	Multi Criteria Decision Making
NAG	Nesterov Accelerated Gradient
RF	Random Forest

RMSPROP	Root Mean Square Propagation
RNN	Recurrent Neural Network
ROS	Robot Operating System
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USB	Universal Serial Bus

LIST OF SYMBOLS

b_i	Offset vector of i-th layer
$downsample$	Down-sampling function
F_i	I-th layer's CNN feature map
s	Pooling size
W_i	I-th layer's weight
ϕ	Rectified Linear Unit (ReLU) activation function
η	Learning rate
β_1	Hyperparameters

CHAPTER 1: INTRODUCTION

With the rapid evolution of technology, robotics has moved from laboratories to almost all aspects of industry, education, health, agriculture, and life in general [Niloy et al., 2021]. The contributions of robotics in relevant areas are remarkable in terms of mass production, safe operation, quality of education, patient satisfaction, and so on. Due to their abilities, mobile robots are one of the most rapidly growing technologies in the world today [Stefek et al., 2020]. Mobile robots can be categorized into two groups according to their operational environment: indoor and outdoor mobile robots. [Yasuda et al., 2020]. Outdoor mobile robots are used in a variety of contexts„ such as agriculture, the military, search-and-rescue, and safety [Rubio et al., 2019]. Due to their high capabilities and human-friendly interfaces, indoor mobile robots have become increasingly prevalent daily. Indoor mobile robots are capable of performing many tasks in a given indoor environment, including cleaning rooms [Muthugala et al., 2020], providing efficient human services [Diddeniya et al., 2019], and serving meals [Guan et al., 2021]. However, indoor mobile robots' dexterous and safe locomotion becomes challenging on surfaces with varying features. Indoor mobile robots require environmental data to adjust their movements intelligently and react to a changing environment. Therefore, timely recognition of surface types is crucial for mobile robots to carry out a given mission successfully [Bai et al., 2019].

The nature of surfaces varies incredibly in ordinary indoor areas. Usually, indoor floors are flat and thus easy to traverse. However, certain areas could be slippery or bumpy, which may result in difficulties in the mobility of robots or hazardous incidents. Therefore, knowledge of the floor type can aid mobile robots in traversing (moving) and adapting their movement according to the surface to avoid undesirable incidents [Xue et al., 2022]. For instance, if the mobile robot identifies surfaces as wood floors, it can move at high speed, as the wood floor is comparatively simple and safe for movement. Certain surfaces, like carpets, are uneven and lofty, which affects movement; thus, mobile robots should slow down their speed. Therefore, the methods for estimating the current or forthcoming floors' characteristics significantly contribute to the motion capabilities of indoor mobile robots. Surface classification can be classified into two main groups based on different sensing techniques: contact-based classification, which

includes vibration and touch, and contactless classification, which mainly makes use of vision and sound.

1.1 PROBLEM STATEMENT

Indoor mobile robots' dexterous and safe moving becomes challenging due to the variety of surface types. In ordinary indoor areas, surface characteristics can greatly vary. Generally, indoor surfaces are flat and simple to move around. However, particular areas can be slippy or uneven, which could make the mobile robots' movement difficult and lead to hazardous incidents. In the context of this, to adapt the movement of the robots according to surface type, avoid undesirable incidents, assign tasks according to the floor type, etc., surface recognition is a crucial problem that needs to be solved.

1.2 RELATED LITERATURE

The classification of surface type or terrain using outdoor robots has gained attention in recent years because of the growing utilization of Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV) in both civilian and military applications; however, as compared to outdoor robotics, the research for indoor surface classification requires more attention. Researchers in prior studies used sensors, cameras, or a hybrid system (sensor and camera both) to detect the surface type. For example, [Tick et al., 2012] classified the indoor surface using data collected from an inertial measurement unit (IMU) connected to the robot. Unlike other sensor-based studies, they used additional attributes such as velocity and acceleration to construct a dataset of 800 features, which were later reduced by exploiting a sequential forward floating feature selector. Then, they used Linear Bayes to classify surfaces into five types (carpet, terrazzo, linoleum, ceramic tiles-A, ceramic tiles-B) and obtained an accuracy of 90% for a robot moving for 20 minutes.

Similar to previous work, [Bermudez et al., 2012] utilized vibration data obtained by an IMU installed on a legged robot. The authors also incorporated magnetic encoders and back-EMF sensors to generate motor control data for better surface classification. They used 75% of the collected data to train Support Vector Machine (SVM) and attained an overall accuracy of 93.8% in the test phase to distinguish three types of surfaces

(carpet, tile, and gravel). [Kertész, 2016] performed several sensing modalities (ground contact force sensor, motor force sensor, infrared sensor, accelerometer, etc.) on the Sony ERS-7 mobile robot to identify six types of indoor terrains with the help of a random forest (RF) classifier. Besides this, a fast Fourier transformation scheme was exploited on the obtained data and the later system was tested. The study achieved an overall accuracy of 94% while moving the robot barefoot and wearing socks.

On the other hand, [Giguere and Dudek, 2011] used a low-speed wheeled robot with an accelerometer to train an artificial neural network (ANN) classifier in order to recognize ten types of outdoor and indoor surfaces. They analyzed acceleration patterns with eight extracted features in the time and frequency domains to obtain an accuracy of 94.6% and 89.9% for windows of 4 seconds and 1 second, respectively. Unlike others, they distinguish between tiled and untiled linoleum surfaces. Instead of using motion or vibration sensors, [Bosworth et al., 2016] utilized touch sensors to measure surface friction and impedance to enable the MIT Super Mini Cheetah robot to move over variable terrains with better locomotion. Previous studies used wheeled or legged robots with different combinations of sensing modalities to classify indoor surfaces based on vibration. However, even an increasing number of sensors does not significantly contribute to achieving higher performance. Moreover, most of these works employed traditional machine learning algorithms, whereas vision-based modalities with advanced deep learning-based models could be more effective for surface classification. In addition to vibration and motion sensing modalities, researchers have also leveraged vision-based data or hybrid datasets (vibration and images) to classify surfaces and terrains.

For instance, [Weiss et al., 2008] utilized imaging data and vibration measurements to classify 14 types of surfaces, including mix grass-gravel, mowed grass, medium grass, short grass, small bushes, dirt, clay, circular paving, quadratic paving, tiles, coarse gravel, fine gravel, asphalts, and indoor surfaces. They first trained an SVM model on the given datasets individually and later broadened the research by combining vibration data with imaging data for better classification. It is noted from the conclusion that the SVM outperformed the models trained on individual datasets, achieving an overall accuracy of 87.04%. Even though they used the camera to capture the images, the

paper focuses only on outdoor surface classification, as they grouped all types of indoor surfaces into one label ('indoor'). In contrast, we aim to classify indoor surfaces into three different labels.

Similarly, [Kurobe et al., 2021] used audio features beside images to identify indoor and outdoor surfaces. They clustered surface types by training the CNN-based ResNet50 model using audio-visual modalities with the help of a microphone and camera installed beneath the mobile platform. Their developed scheme achieved an overall accuracy of 80%. Nonetheless, including audio modality does not enable the model to outperform vibration-based classification models. Therefore, there is a need to suggest an accurate indoor surface classification system with minimal equipment and computational cost. [Guan et al., 2022] proposed a framework that utilizes visual and inertial perception by using camera and IMU sensors to classify multiple outdoor and some indoor surfaces, such as carpet and tile, for mobile robot navigation. The presented framework uses a CNN-based EfficientNet-B0 model to extract the image features. The obtained results demonstrate the efficacy of the proposed model for surface classification tasks. However, the indoor surface types in our study are more diverse, including a wide range of carpet, tile, and wood types in various colors and patterns. Additionally, our dataset consists of samples collected from different indoor environments, unlike previous studies, which focused on a single site.

1.3 CONTRIBUTION OF THE THESIS

Previous studies on surface classification have certain limitations. Firstly, the generalization of the trained classification networks is not specified, whereas models were trained over finite samples that may cause over-fitting or underfitting. Secondly, most works did not present the time analysis to detect the sudden changes in the surface. Thirdly, they lack a discussion of model efficiency regarding training time. Fourthly, the presented results include limited performance evaluation metrics. Fifthly, most studies exploited specific machine learning algorithms. Still, they lack a comparative analysis of the proposed model with state-of-the-art models, which may perform better than the suggested ones. This study aims to improve surface recognition to stabilize the mobile robot's movements so that it performs the given task more comprehensively. Besides exploiting pre-trained state-of-the-art deep learning models, the study proposed a

modified MobileNetV2 to classify the surface efficiently. The key contributions include:

- Generation of a unique dataset that is composed of surface images captured in different environmental conditions.
- Unlike prior studies, our new dataset includes carpet surfaces besides wood and tiles.
- Comparison and analysis of the pre-trained deep learning models for a surface classification task on the new dataset.
- Proposing a modified MobileNetV2 to establish its impact on the classification performance of robotic systems with limited computational capacity.
- Comparison and analysis by examining the effect of several optimizers (SGD, RMSProp, Adam, Adadelta, Adamax, Adagrad, and Nadam) to distinguish the most efficient network for the problem of surface recognition.
- Providing a systematic evaluation of each utilized model using various optimizers based on accuracy, precision, recall, and F1-score.
- The performance analysis of the MobileNetV2 Proposed model was tested in real-time using the Kobuki robot by creating 11 different scenarios in different environments such as a home, office, and hallway consisting of carpet tiles and wood.
- Indoor robots, Kobuki and Burger, were tested by considering the parameters such as IMU data for each surface at different speeds, the margin of error in cm when reaching the target, and completion time.
- Determining which robot performed better on which ground by using the Analytic Hierarchy Process method, which is a type of Multi-Criteria Decision Making, based on the parameters obtained during the test.
- Task assignment with multi robots according to surface type.

1.4 APPROACH

The main goal of this thesis is to develop a system that can classify the surface types in real time and assign tasks to heterogeneous robots according to the surface type. The system consists of four main parts: generating a new dataset, applying several Deep Learning based methods as well as proposing a lightweight MobileNetV2 model, performance analysis of the Kobuki and Burger mobile robots, and finally utilizing mobile robots and ensuring communication between them.

DL-based techniques heavily rely on datasets, accurate and precise results cannot be obtained without reliable and useful data. Considering this, we generated a dataset that includes various types of carpet, tiles, and wood images captured in different environments. After the collection part, the dataset was prepared for further processing.

Once the dataset was ready to use, DL-based models were applied to perform surface classification. To find the most accurate and lightweight model state-of-the-art CNN models as well as the proposed model were performed with several different optimizers. Then, each model's performance was tested on a main computer in terms of accuracy, precision, recall, and F1 score. From each model, we obtain pre-trained weights, which were later used for surface classification in real time on both Kobuki and Burger robots. Figure 1.1 illustrates the surface classification system applied in this thesis.

The Kobuki and Burger robots have been prepared to conduct the experiments by installing all the packages and components on them. The connectivity between robots and computers was facilitated by the Robot Operating System (ROS). ROS packages were utilized for the real-time classification, robot movement, and connection of the components. The communication between robots was done via Wi-Fi protocols.

The performances of the Kobuki and Burger robots on different surfaces were examined in terms of angular deviation, time to reach the desired point, and precision. The measurement of angular deviation was taken by the Inertial Measurement Unit (IMU) sensor. We defined start and end points with a length of 5 meters on each surface. Later, we moved both robots across the same scenario. By comparing their performance, we

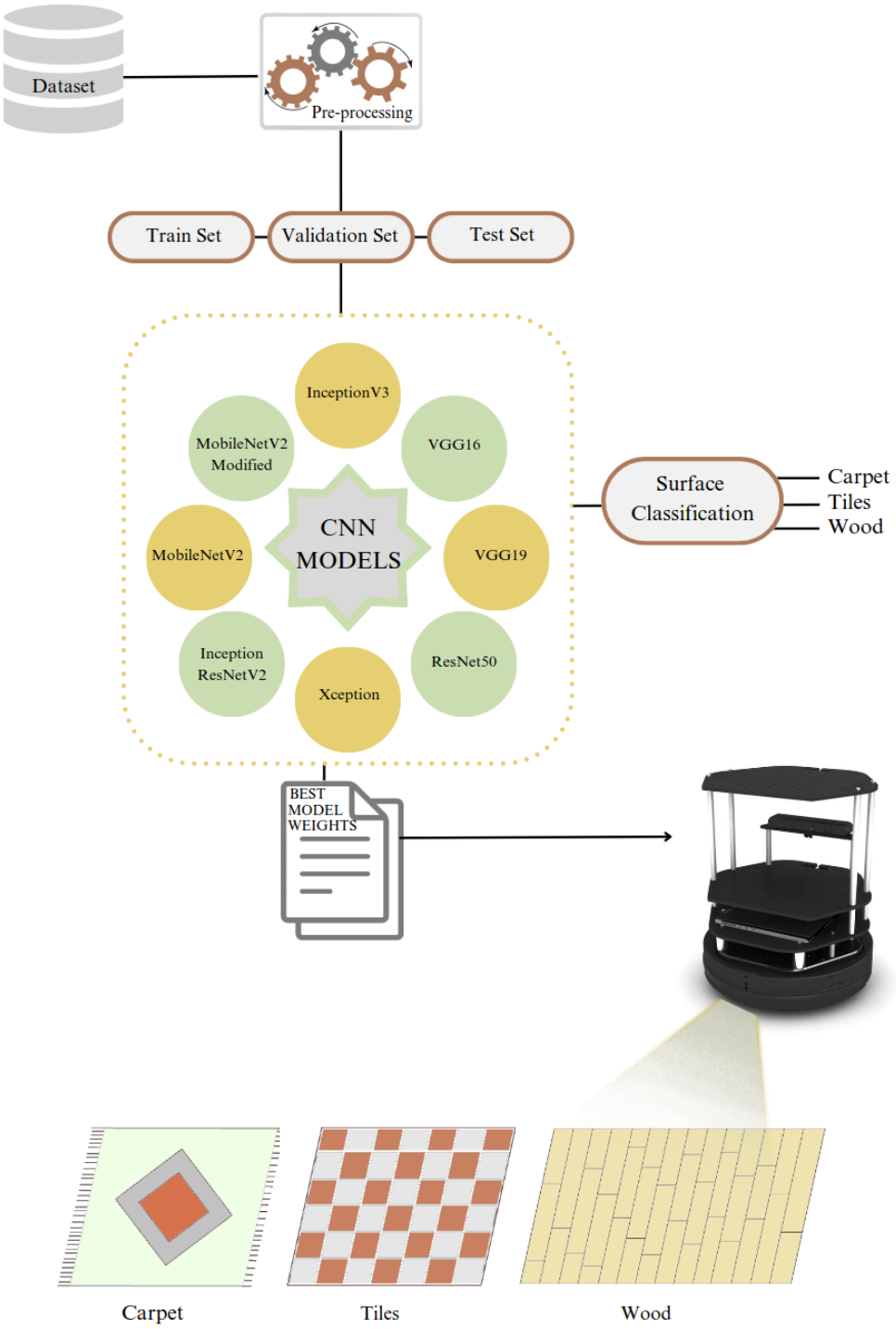

Figure 1.1: Problem Illustration.

determined which robot performed better on each surface. According to this knowledge, tasks were assigned to robots.

Furthermore, the proposed approach was tested in different indoor environmental scenarios. The optimized model weights obtained are uploaded to both robots, so surface

classification was done in real-time and according to the surface type tasks were assigned to the robots automatically. Finally, we performed task assignment according to the surface type experiments. The obtained results show that task assignments were carried out successfully.

1.5 OUTLINE OF THE THESIS

The remaining four chapters of the thesis are organized as follows. Chapter 2 provides a comprehensive review of the methodologies and algorithms applied at each stage to achieve the surface classification goal. As surface classification utilizing the DL approach is the primary focus of this chapter, it elaborates on the fundamental capabilities of CNN models and discusses a few optimizers. Furthermore, it presents the newly generated dataset, and provides detailed insights into the experiments and their corresponding results. In Chapter 3, the method used for the performance analysis of the robots is described. Chapter 4 demonstrates the experiments, while Chapter 5 concludes the thesis.

CHAPTER 2: CNN BASED SURFACE CLASSIFICATION

This section presents a detailed description of the dataset generated in this thesis, along with the selection and architecture design of pre-trained CNN models and proposed model, also the application of various optimization algorithms. It specifically focuses on the generation of a new dataset, pre-processing steps, the implementation of CNN-based models, and the selection of the appropriate optimization algorithms. The illustration of the applied and proposed model in this section is presented in 2.1.

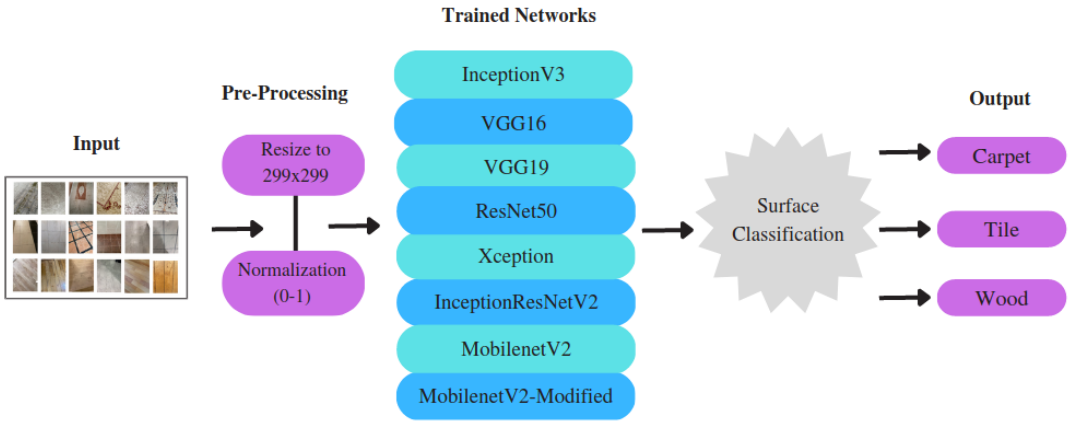

Figure 2.1: Demonstration of the System for Surface Classification.

2.1 DATASET

Datasets play a significant role in DL-based techniques, and accurate and precise results cannot be achieved without consistent and meaningful data. As there is a lack of an open-source indoor surface image dataset, we generated a new dataset that includes three distinct floor types: carpet, tiles, and wood. We captured dataset samples with cameras in various indoor environments and lighting conditions. Before building the dataset, we analyzed the overall dimensions of the indoor robots. Thus, while generating the dataset, we took images from different angles to ensure convenience for the indoor mobile robot's camera position. According to our literature review, the generated dataset is one of the most comprehensive datasets used in such studies. In particular, to prevent the lack of datasets in which the diversity of indoor surfaces is limited, samples in our dataset were captured in more than 20 indoor environments, including various carpet, tile, and wood floors [Demirtas et al., 2023]. In Figure 2.2, some samples from the generated dataset are presented. The generated dataset is publicly

available in [Demirtas et al., 2023].

Figure 2.2: Samples from the Different Surface Types.

The generated dataset consists of a total of 2081 samples, of which 870 samples are carpet, 638 tiles, and 573 wood floors. All the images are RGB, and the size of the collected images varies, but before feeding to the CNN models, samples are resized to an equivalent dimension. Detailed information about the generated dataset can be found in Table 2.1.

Table 2.1: Details of the Generated Dataset

Dataset	**Carpet**	**Tiles**	**Wood**	**Total**
Train	698	510	457	1665
Test	86	64	58	208
Validation	86	64	58	208
Total	870	638	573	2081

2.2 SURFACE CLASSIFICATION USING CNN

Convolutional Neural Networks (CNNs) are a Deep Learning (DL) based approach that is extensively adopted for image recognition and classification tasks in various areas [Huang et al., 2017, Szegedy et al., 2015]. Instead of manually extracting certain features, as is the case with traditional methods, CNNs can automatically learn specific features from original images without the need for human supervision. The convolutional layer, pooling layer, and dense layer are the three basic layers that make up a CNN architecture. The descriptions of each layer are given in the following.

2.2.1 Convolutional Layers

The most crucial part of CNN is the convolutional layer, that utilizes various convolution kernel sizes to obtain the particular features of a given sample. A set of feature maps can be obtained by applying a convolutional layer in a given image a few times. Assuming that F_i represents the CNN feature map for the ith layer, F_i is given as:

$$F_i = \phi(F_{i-1}W_i + b_i) \tag{2.1}$$

where F_i indicates the current network layer's feature map, F_{i-1} represents the previous layer's convolution feature, W_i represents ith layer's weight, b_i is the offset vector of the ith layer, and ϕ represents the rectified linear unit (ReLU) activation function.

2.2.2 Pooling Layers

After applying a convolutional layer, pooling layers reduce the feature dimensionality, thus drastically minimizing the computational complexity. In addition, pooling layers effectively control the risk of overfitting. The output feature of the hth local receptive field in the lth pooling layer can be calculated as follows:

$$x_h^l = downsample(x_h^{l-1}, s) \tag{2.2}$$

where *downsample* indicates the down-sampling function, x_h^{l-1} represents the feature vector of the previous layer, and s denotes the pooling size.

2.2.3 Dense Layers

Finally, all the features extracted by previous layers are collected in the fully connected layer, thus, surface classification can be performed. The Softmax function generally performs class prediction with all the gathered features. The Softmax function can be expressed mathematically as:

$$softmax(z)_j = e^{z_j} / \sum_{k=1}^{K} e^{z_k}, \forall j \in \{1, ..., K\} \tag{2.3}$$

where K denotes the dimension of vector z.

2.3 PRE-PROCESSING

Carrying out a few pre-processing steps before feeding the data into DL models plays a critical role in improving performance and feasibility. To meet the fixed-size input requirement of applied and proposed CNN models, we pre-processed the input size of the samples to 299x299 pixels. We then performed min-max normalization, which reduced the values of a given set to a ratio between 0 and 1. Min-max normalization is expressed by Equation 2.4 [Rajendran et al., 2020]:

$$I_{out} = (I_{in} - Min)\frac{newMax - newMin}{Max - Min} + newMin \tag{2.4}$$

2.4 CNN BASED MODELS

In this thesis, several CNN-based models were used to perform the surface classification tasks, including InceptionV3, VGG16, VGG19, ResNet50, Xception, InceptionResNetV2, MobileNetV2, and the proposed MobilenetV2-Modified model. The description of each model is presented in detail below.

2.4.1 InceptionV3

The InceptionV3 is a CNN-based DL model that is used for image recognition tasks [Szegedy et al., 2016]. It is an advanced version of the main Inception V1 model. In the InceptionV3 model developed by Google, a group normalization and a fully connected layers have been added in order to make the network more efficient. This architecture is widely utilized for image classification and has demonstrated good results on the ImageNet dataset. The improved version aims to perform well even on applications with limited computational costs. For example, a 3×3 convolution is split into 1x3 and 3x1 convolutions, which allows for more effective extraction of specific details present in the image and reduces the numerical parameters of the model, resulting in shorter

training times. The total depth of the model is 47 layers. Despite its more profound architecture than previous versions, InceptionV3, with the new factorization ideas, reduces the parameters without decreasing the network performance.

2.4.2 VGG16

VGG16 as a CNN-based model, was ranked among the top 5 models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015] competition on the ImageNet dataset in 2014 with a 7.3% error rate [Simonyan and Zisserman, 2014]. The architecture comprises 16 layers with approximately 130 million parameters, applying 3×3 filters. Instead of using a lot of hyper-parameters, VGG16 uses on 2x2 filtered max-pool layers and 3x3 filtered convolution layers, which have a stride of 1. The architecture's fundamental components are convolutional layers with varying depths, followed by three fully connected layers. The first and second dense layers contain 4096 neurons, while the third has 1000 neurons. Softmax is the last layer where classification is performed.

2.4.3 VGG19

The overall concept of the VGG19 architecture is the same as VGG16, except for the adoption of three additional convolutional layers. In this case, the first 16 convolutional layers extract features, while the following three fully connected layers are used for classification. There are five groups of feature extraction layers, each of which is followed by the max pooling layer. VGG19 model's input size is the same as VGG16's 224 x 224 pixels.

2.4.4 ResNet50

The classification accuracy of deep CNN models increases correspondingly with the number of network layers. However, as the depth of network increases, training and test error rates of the models also increase. This case is referred to as the "vanishing gradient". To address this problem, the Residual Network was developed [He et al., 2016]. ResNet50 deploys a method referred to as skip connections, allowing the network to link directly to the output by skipping several training layers. The underlying architecture of ResNet50 is inspired by the VGG architecture, consisting of five phases,

each with an identity block and a convolutional block. Three convolutional layers compose each block. Downsampling is performed using pooling layer. The network is finalized with fully connected layers.

2.4.5 Xception

As an inspired version of the Inception model, the Xception architecture was introduced by [Chollet, 2017] in 2017. The Xception model is composed of deep and comprehensive convolutional layers that operate in a collateral manner. Thus, the feature extraction process of the network is realized with a total of 71 convolutional layers. In Xception, the convolutional layers are arranged into modules and encircled by linear residual connections, making them stacks of depthwise separate convolutional layers covered by residual connections. This makes the architecture very simple to describe and perform, unlike InceptionV2 or V3, which are substantially more difficult to specify. The pre-trained model, which provides excellent efficiency in image recognition, was trained on the Imagenet dataset using millions of samples.

2.4.6 InceptionResNetV2

The InceptionResNetV2 module is a combination of Inception and residual networks, showing promising results in image classification and object detection tasks [Szegedy et al., 2017]. From a particular view, each InceptionResNet module can be demonstrated as a tiny CNN. These small CNNs, together with additional network layers like pooling and convolutions, compose the main architecture of the InceptionResNet network. The architecture of the InceptionResNet module consists of two main parts: shortcut connections and residual blocks. Input features are directly mapped to output features by the shortcut connection, so residual blocks estimate the residual function. Finally, the module output is composed of the corresponding map of input features as well as the output of the residual block.

2.4.7 MobileNetV2

The MobileNetV2 model is utilized in many computer vision applications, but mainly it offers high robustness in object recognition tasks. The model increases the performance of the existing CNN models on mobile and computationally limited devices

across various benchmarks, activities, and model sizes [Sandler et al., 2018]. On the main architecture of MobileNetV2, bottleneck levels, inverse residual structures, and connections are presented. This methodology makes real-time classification possible on computationally limited devices or smartphones. More detailed information about the fundamental architecture of MobileNetV2 can be found in [Sandler et al., 2018].

2.4.8 MobileNetV2-Modified

The fundamental MobileNetV2 architecture inspires our proposed modified MobileNetV2 model. Once the core convolutional architecture has extracted the input images' features, global average pooling layer is utilized to minimize the size of the extracted features. So, the precition is done in the fully connected layer. Original MobilenetV2 model has a lower computational cost than other CNN-based models. Thus, this makes the MobilenetV2 model more effective in mobile robot-based applications. In this thesis, since the surface classification performance of the mobile robot was analyzed, it is more consistent to modify the MobilenetV2 model compared to other models in terms of computational cost. While the original MobileNetV2 model has an average weight of around 40 MB, which is relatively small, its architecture is changed to reduce the mobile robot's computational cost and make it work more efficiently. To achieve this, we replace the last 11 layers before the fully connected layers of the original MobileNetV2 with 2D-convolutional layers with 16 filters with a kernel size of 5x5, ReLu as activation, and the same padding. Thus, the number of blocks was reduced from 17 to 16, as shown in Figure 2.3 and 2.4. The convolutional layer is followed by a Global Average Pooling layer and a Dropout layer with a value set to 0.2. Then, two dense layers are added, having a size of 2096 and 3, respectively. Softmax is used for activation to classify the three types of surfaces correctly. The size of the extracted features is minimized using the Global Average Pooling layer, and overfitting was prevented by adding a new Dropout layer to the modified architecture. Apart from removing blocks, adding dropouts, and building a convolution layer, we made a significant change to the proposed model by making the last 23 layers trainable (Figure 2.4), whereas in the original MobileNetV2, all layers are frozen except for the fully connected and global average pooling layers. In the MobileNetV2 model, the frozen layers are trained on the ImageNet dataset, while the trainable layers are trained with a custom dataset. By doing so, the number of training parameters is reduced, thus significantly reducing the

computational time compared to the original MobileNetV2, which will be explained further in the comparison section.

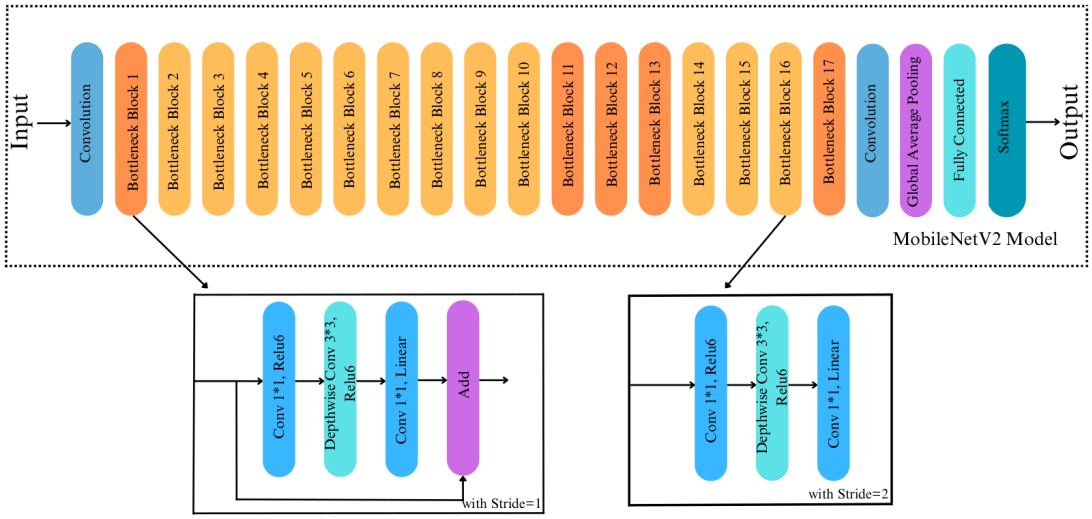

Figure 2.3: Original Architecture of MobileNetV2.

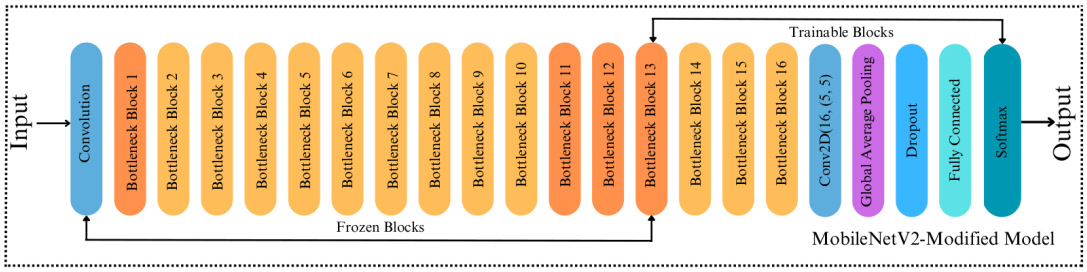

Figure 2.4: Architecture of MobileNetV2-Modified.

In Algorithm 1, firstly, the surface dataset was loaded (line 1). Secondly, normalization was performed to reduce the computational cost (line 2). The MobileNetV2 model architecture was defined (line 3). Following, the last 11 layers of the original MobileNetV2 model was deleted (line 4-6) and replaced with a convolutional layer, a global average layer, a dropout, and two dense layers (line 7-13). After the modifications to the architecture, we made the last 22 layers trainable (lines 14-16). The normalized dataset was separated into training, validation, and testing (line 17). The modified model was trained (line 18) and tested (line 19). Finally, the prediction result was returned (line 20).

Algorithm 1 Pseudo Code of the Proposed Model

```
ImP=Load(Dataset)
ImNorm=(ImP min(ImP))/(max(ImP)min(ImP))
Architecture ← {MobileNetV2}
for i in |Architecture| do
    editArchitecture=[i, −11] delete
end for
model define do
addlayer1←conv2D (editArchitecture)
addlayer2←globalAveragePooling (editArchitecture)
addlayer3←dropout (editArchitecture)
addlayer4←dense (editArchitecture)
addlayer5←dense (editArchitecture)
modifiedModel← (addlayer1, addlayer2, addlayer3, addlayer4, addlayer5)
for j in |modifiedModel| do
    [j, −22] unfreeze layers
end for
f_train, f_validation, l_test ← ImNorm
training ← modifiedModel (f_train, f_validation)
score ← (l_test, training)
return score
```

2.5 OPTIMIZERS

Optimizers regulate the learning speed and the neural network weights to provide the best model performance and reduce the losses experienced during training. In other words, the optimization algorithms minimize the loss during training and ensure accurate dataset training. The optimizer algorithms applied in this thesis are briefly explained below.

2.5.1 SGD (Stochastic Gradient Descent)

Stochastic Gradient Descent [Robbins and Monro, 1951] and its variations are frequently utilized in deep learning. SGD is different from the classic vanilla gradient descent technique in that it computes the gradient of the objective function. It updates the parameters on subsets of the training data instead of the complete training set. The main goal of SGD is to minimize computational costs. It is worth noting that gradient descent is a computationally intensive process by itself, especially when there is a large training set. The parameter update of SGD is done as follows:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta_{t-1}) \tag{2.5}$$

where $\theta_t \in R^d$ is the parameter vector during iteration t, the learning rate is represented by η, $J(\theta_{t-1})$ is the loss function defined by the model's parameters θ_t at the t-th iteration, and $\nabla_\theta J(\theta_{t-1}) = \partial J(\theta_{t-1})/\partial\theta$ represents the gradient of the loss function to parameters at the $t-1$th iteration.

2.5.2 Adam (Adaptive Moment Estimation)

The advantages of Adagrad and RMSprop optimizers are combined in Adam, which calculates the learning rates adaptively for various variables. It has a low memory demand, high computational efficiency and calculates the learning rates for each parameter. Momentum is directly integrated with Adam to estimate the gradient's first-order moment. To account for the initialization at the origin, Adam applies bias corrections to the estimations of the first-order and second-order moments [Kingma and Ba, 2014]. To estimate the moments, Adam employs the exponential moving average, which is computed on the gradient evaluation concerning the current mini-batch. The exponential decay rates of these moving averages are controlled by the hyperparameters β_1 and β_2, which are usually set close to 1. β_1 and β_2 are usually set near 1. The estimate of the gradient's mean, m_t, is calculated as:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.6}$$

where g_t is the gradient on the current mini-batch at step t. Subsequently, the estimate of the gradient's uncentered variance, v_t, is calculated as:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.7}$$

In the first stage, m_t and v_t are smoothly biased to starting value. Therefore, bias

correction must be calculated for both the first and second moments, so the following steps are performed:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.9}$$

Finally, the parameters are updated as follows:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_{t-1}} + \epsilon} \hat{m}_{t-1} \tag{2.10}$$

where η is the learning rate and ϵ is a small value to avoid division by zero.

2.5.3 RMSProp (Root Mean Square Propagation)

The RMSProp algorithm was proposed by Geoffrey Hinton [Hinton et al., 2012]. The RMSProp optimizer have common points with gradient descent algorithm. By employing a moving average of the squared gradient to normalize the gradient by considering the size of recent gradient descents, RMSProp attempts to address the drastically decreasing learning rates of Adagrad. Therefore, the algorithm progresses horizontally, with larger steps converging faster as the learning rate increases [Mukkamala and Hein, 2017]. The moving average of the squared gradient is maintained as follows:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E[g^2]_{t-1} + \epsilon}} g_{t-1} \tag{2.11}$$

2.5.4 Adagrad

Adagrad is an optimization technique that modifies the learning rates of the model parameters on an individual basis [Duchi et al., 2011]. The learning rate of the parameters with the biggest partial derivative of the loss decreases quickly, whereas the learning rate of the parameters with smaller partial derivatives decreases more slowly [Goodfellow et al., 2016]. This is accomplished by using all the previous squared values of the gradient. The update for each parameter θ_i in every iteration t is expressed as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\eta}{\sqrt{G_{t-1,ii} + \epsilon}} \cdot g_{t-1,i} \tag{2.12}$$

Here, ϵ is an expression that prevents division by zero, and the objective function's gradient with respect to the parameter θ_i at iteration $t-1$ is denoted with g_{t-1}:

$$g_{t-1,i} = \nabla_\theta J(\theta_i), \tag{2.13}$$

where $g_{t-1,ii}$ is a diagonal matrix where every diagonal element i, i is the sum of gradients squares in respect to θ_i up to iteration $t-1$. Element-wise vector multiplication between G_{t-1} and g_{t-1} is represented with $\odot$, and vectorization update is done as follows:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_{t-1} + \epsilon}} \odot g_{t-1} \tag{2.14}$$

2.5.5 Adadelta

The primary goal of the AdaDelta method is to address the two main issues with AdaGrad: the constant decay of learning rates during training and the demand for manually selecting global learning rates. To achieve this, AdaDelta limits the past gradient window to a fixed size rather than adding up the sum of all squared gradients

over time [Zeiler, 2012]. The update of Adadelta is as follows:

$$\theta_t = \theta_{t-1} - \frac{RMS[\Delta\theta]_{t-2}}{RMS[g]_{t-1}} g_{t-1} \tag{2.15}$$

where $RMS[g]_{t-1}$ represents the RMS error criterion of the gradient $[g]_{t-1}$:

$$RMS[g]_{t-1} = \sqrt{E[g^2]_{t-1} + \epsilon} \tag{2.16}$$

and the running average at the $t-1$th iteration is represented with $E[g^2]_{t-1}]$, which is calculated using the previous average and the exist gradient:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_{t-1}^2 \tag{2.17}$$

where γ denotes a fraction that is identical as the momentum term. Following this, $RMS[\Delta\theta]_t$ represents the error criterion of $\Delta\theta^2$, and the running average $E[\Delta\theta^2]_t$ of $\Delta\theta_t$ is acquired by:

$$E\left[\Delta\theta^2\right]_t = \gamma E\left[\Delta\theta^2\right]_{t-1} + (1-\gamma)\Delta\theta_{t-1}^2 \tag{2.18}$$

2.5.6 Adamax

Adamax is an extension of Adam. First, Adamax calculates the gradients at time step t concerning the stochastic objective. Then, it calculates an exponentially weighted infinity norm along with a biased first-moment estimate to update model parameters [Kingma and Ba, 2014]. Adamax's parameter update rule is as follows:

$$u_t = max(\beta_2 v_{t-1}, |g_t|) \tag{2.19}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t \tag{2.20}$$

2.5.7 Nadam

Nadam uses the accelerated gradient of Nesterov to modify Adam's momentum component. Thus, the goal of Nadam is to increase the speed of convergence of the models [Dozat, 2016]. Similar to the Adam algorithm, the first and second-moment variables are updated after computing the gradients. Following this, corrected moments are calculated so that the parameters can be updated. The parameter updates are expressed as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\left(\beta_1\hat{m}_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right) \tag{2.21}$$

2.6 LIBRARIES

The libraries that were used throughout the thesis to apply algorithms and conduct experiments are described below.

2.6.1 TensorFlow and Keras

Two well-known open-source libraries that are widely utilized in the deep learning and artificial intelligence fields are TensorFlow and Keras. Developed by Google, TensorFlow offers a complete collection of tools and resources for creating and deploying AI models. It provides an adaptable and scalable infrastructure that enables researchers to build neural networks, manage large datasets, and effectively improve model performance. TensorFlow's strength rests in its capacity to make use of both Central Process Units (CPU) and Graphics Processing Units (GPU), enabling high-performance computing and faster training times. On the other side, Keras, which was formerly a standalone library, now integrated into TensorFlow. With its straightforward and user-friendly API, users can quickly create and set up their models with minimal code. In addition, Keras offers a wide range of pre-built layers, activation functions, and

optimization algorithms. Keras supports various network architectures such as CNN, Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and more. The tools for data pre-processing and model evaluation are also included in Keras, which makes it a complete solution for deep learning applications. In our thesis, we used both TensorFlow and Keras libraries in building the models, training, and testing [Abadi et al., 2015].

2.6.2 Scikit-learn

Scikit-learn, also referred to as sklearn, is a well-known Python machine learning and deep learning library. The library offers a wide range of techniques and tools including classification, clustering, and model selection. Sklearn also provides extensive functionality for model evaluation, data preprocessing, and feature extraction, ensuring a comprehensive pipeline for developers. We utilized the Sklearn library in obtaining the confusion matrices of the trained CNN models [Pedregosa et al., 2011].

2.6.3 Matplotlib

Matplotlib, as a visualization library, offers users a wide range of plot types, such as line plots, charts, figures, and more. Users by utilizing the Matplotlib library can change the colors, styles, axis labels, and titles on every part of their plots. It ensures precise control over figure size, aspect ratio, and subplot arrangement, thus, developers can create informative visualizations. The results obtained from the CNN models were plotted by utilizing the Matplotlib library [Hunter, 2007].

2.6.4 Seaborn

Seaborn as a Python library, is intended to offer a more advanced user interface and more capabilities for developing visually appealing and informative statistical graphics. It provides a variety of plot forms, such as scatter plots, bar plots, box plots, and more. With just a few lines of code, Seaborn can produce complex statistical visualizations including regression plots, distribution plots, and correlation matrices, which is one of its main advantages. Seaborn library's one of the standout feature is that it can produce complex statistical visualizations including regression plots, distribution plots, and correlation matrices. The confusion matrices in our research are visualized via

Seaborn library [Waskom et al., 2017].

2.7 TRAINING AND TESTING RESULTS OF SURFACE CLASSIFICATION MODELS

In this study, we propose a solution for surface classification tasks and evaluate the performance of the applied and proposed models. The evaluation process for the classification task in terms of training and testing is performed on the AMD Ryzen 9 5900X processor with 64 GB of RAM and the Nvidia GeForce RTX 3080 Ti graphic card. Jupyter Notebook with Python 3.6 is used as an interface. Tensorflow and Keras libraries are used for the implemented models. Matplotlib and Seaborn libraries are utilized for visualization.

To evaluate the model's accuracy, each model is trained and tested on a dataset containing 2,081 images. Dataset was divided into training, validation, and testing. We used 80% of samples for training, 10% for validation , and 10% for testing. Note that test images are selected randomly to predict the surface accurately. The models efficiency and performance was evaluated with four metrics: accuracy, precision, recall, and f1-score. Confusion matrices are demonstrated to analyze the model's predictions in detail. The performance evaluation metric equations are expressed below:

- The proportion of correctly identified samples to all samples is used to measure accuracy. This metric illustrates the classification's level of confidence.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.22}$$

- Precision is a metric that determines the ratio of true positives (TP) to the total of TP and false positives (FP).

$$Precision = \frac{TP}{TP + FP} \tag{2.23}$$

- Recall is the metric that demonstrates the ratio of the true positive to the total of the true positive and false negatives(FN).

$$Recall = \frac{TP}{TP + FN} \tag{2.24}$$

- The F-measure is a metric defined as the harmonic mean of precision and recall.

$$F1_{score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.25}$$

The train and validation accuracy/loss graphs obtained with the best-performing optimizers of each model are demonstrated between Figures 2.5-2.12. When the figures are examined, it is seen that the train accuracy values of the models are above 96%. However, the values of the validation accuracy vary significantly according to the models. For instance, as shown in Figures 2.5-2.7, although train accuracy is high in the InceptionV3 and VGG19 models, validation accuracy could not exceed a specific limit. It implies that there is overfitting in both models. When analyzing the train-validation loss graphs of VGG19 and InceptionV3, it is observed that as the number of epochs increases during training, the train loss value of VGG19 decreases, whereas the InceptionV3 model increases.

Contrary to these models, it is seen that train and validation accuracy values in VGG16 and Xception models converge, and overfitting decreases compared to previous models, except for a few epochs when examining Figures 2.6-2.9. This decrease in overfitting accuracy values is reflected in the graph as an increase in loss values. Although there is oscillation for the InceptionResNetV2 model in Figure 2.10, the train and validation accuracy values are close. The train loss value is already low, and the validation loss confirms this.

Comparing the MobilenetV2 and MobilenetV2-Modified graphs in Figures 2.11 and 2.12, it is evident that the training process of the proposed model is better than the original MobileNetV2. Thus, this positively affects the proposed model's accuracy, precision, recall, and test accuracies.

The performance comparison analysis of the proposed model and applied state-of-the-art DL models for surface classification tasks are shown in Table 2.3. Experimental

results demonstrate that the modified model surpassed the state-of- the-art DL models in terms of precision, recall, f1-score, and test accuracy when trained with the Adamax optimizer, achieving 4% higher accuracy than each model. Afterward, the most effective ones are MobileNetV2, InceptionV3, Xception, and InceptionResNetV2 models, whereas VGG19 shows the lowest performance when considering the results obtained with different optimizers. Among the seven different optimizers, Adamax, Adam, SGD, and RMSProp perform the most effectively on each CNN model, while the others perform less effectively.

When Figures 2.11 and 2.12 are compared, the effect of the modification can be seen quite clearly in terms of the accuracy and loss value. When the train-validation accuracy graph in Figure 2.11 is examined, it is seen that the oscillation is lower, and there is a small difference in accuracy compared to the other models. As seen in Figure 2.12, the training progress of the modified model has been completed consistently. In the train-validation accuracy and train-validation loss graphs, it is seen that the values converge between train-validation accuracy and loss. Thus, overfitting is prevented.

Consequently, the test accuracy is obtained as 99.52%, which is quite high compared to other models, and the loss value is close to 0 in Table 2.3. Therefore, this shows how well the modified model was trained. As seen in Figure 2.13, MobileNetV2 Modified mispredicts only 1 carpet image out of 208 test images in the confusion matrix, which reveals that the test accuracy is almost 100%.

The confusion matrices for each model are shown in Figure 2.13, with each matrix representing the classification of images into 0 Carpet, 1 Tile, or 2 Wood. As shown in Figure 2.13, 86 images of Carpet, 64 images of Tile, and 58 images of wood were tested. Each class was predicted with high accuracy, as indicated by the Confusion Matrix results. When examining the estimation of Carpet for each model, it is observed that the TP value is relatively high and the FP value is low, which is desirable. For instance, when the confusion matrix is examined for tile, the FN value is higher than other classes by reducing the recall value. It was determined as wood instead of being estimated as tile. This is because there are wood-like tiles and tile-like wood images in the section reserved for testing. In contrast, wood appears to be predicted more

accurately. Considering the confusion matrices, the modified model has shown the highest prediction when compared to the applied DL models.

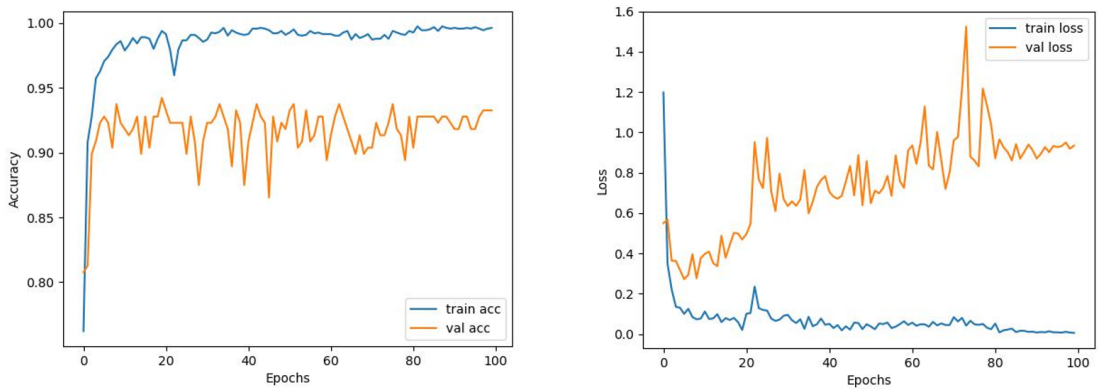

Figure 2.5: InceptionV3 Model's Training and Validation Accuracy-Loss Graphs with Adam Optimizer.

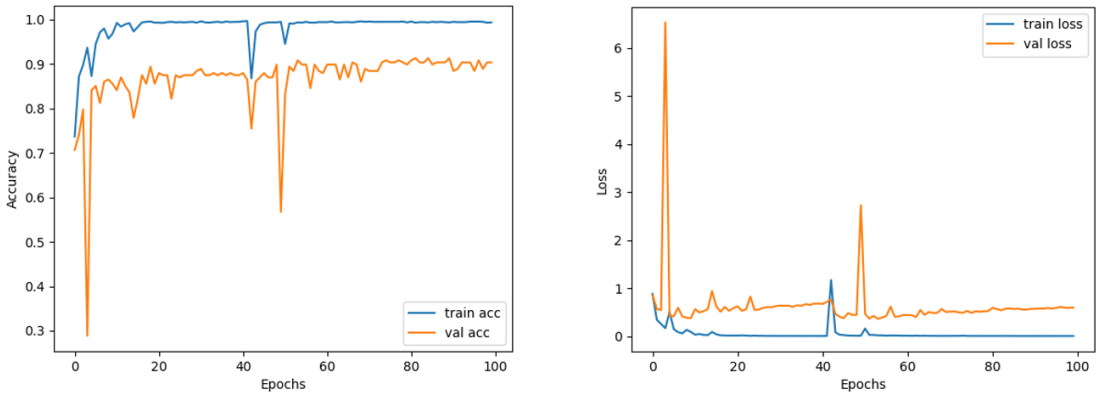

Figure 2.6: VGG16 Model's Training and Validation Accuracy-Loss Graphs with Nadam Optimizer.

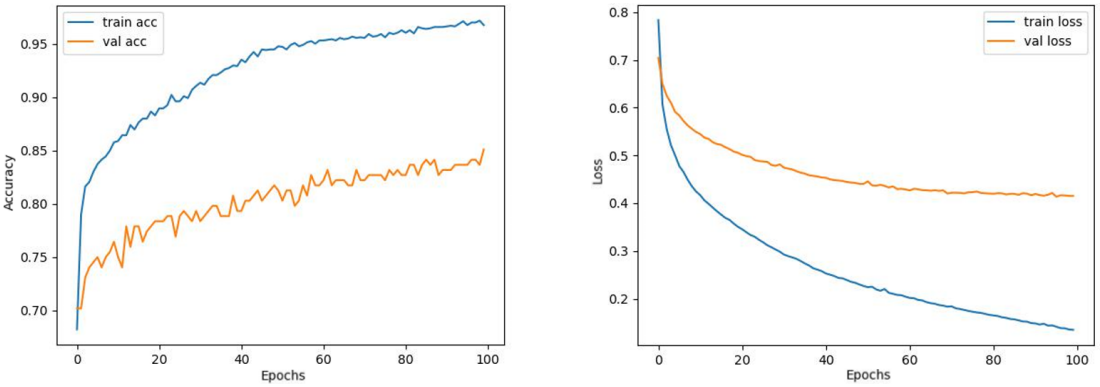

Figure 2.7: VGG19 Model's Training and Validation Accuracy-Loss Graphs with Adamax Optimizer.

Additionally, in this study, the computational time of each model was also measured. Table 2.2 presents the training times and the weight sizes of both the modified model and the state-of-the-art DL models used in the surface classification task. Compared to the other models, the modified model effectively completed the training in less com-

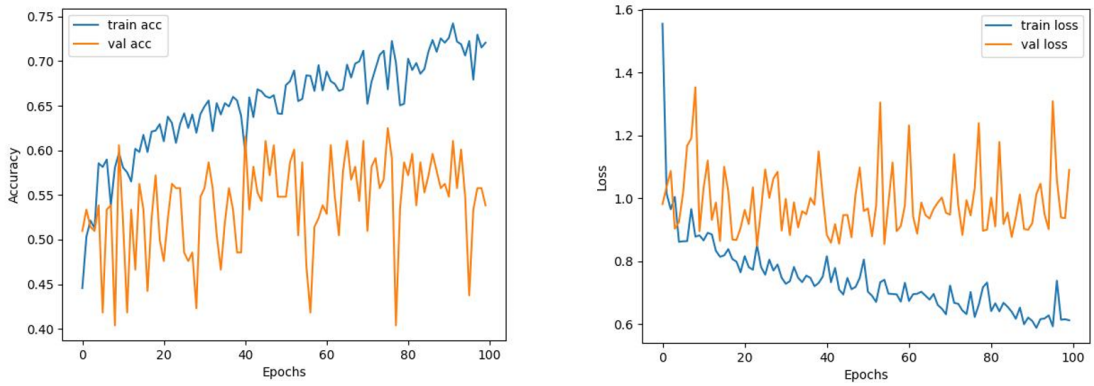

Figure 2.8: ResNet50 Model's Training and Validation Accuracy-Loss graphs with Adamax Optimizer.

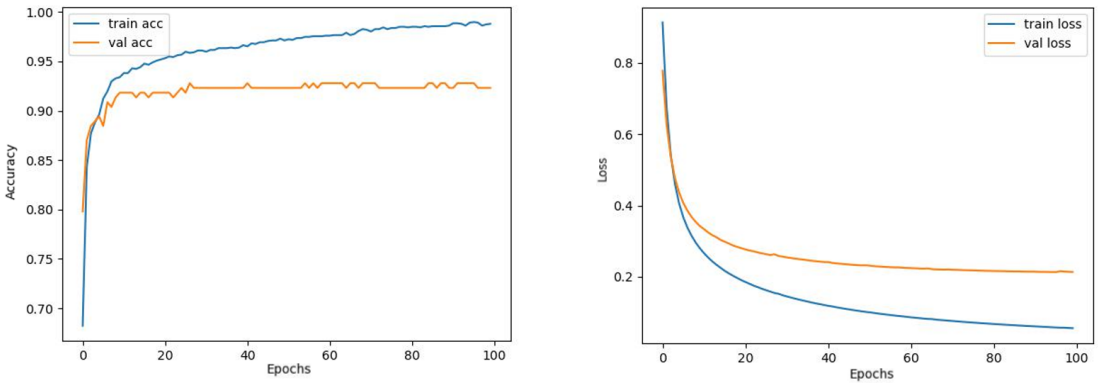

Figure 2.9: Xception Model's Training and Validation Accuracy-Loss Graphs with SGD Optimizer.

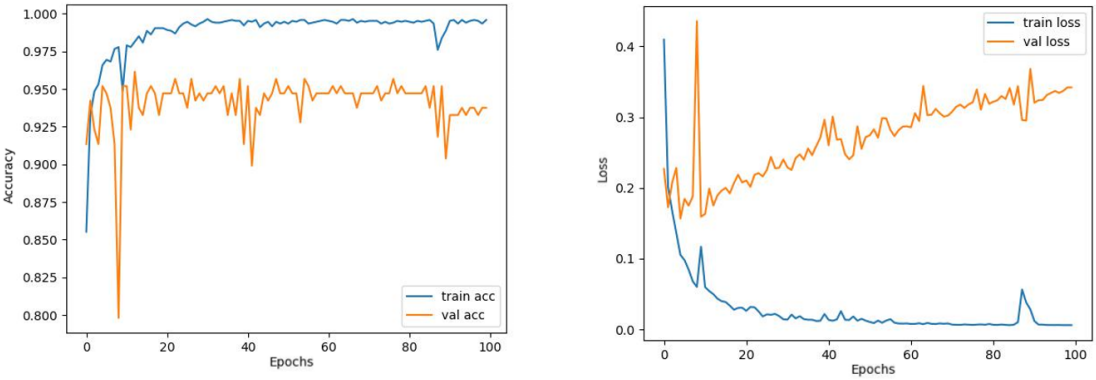

Figure 2.10: InceptionResnetV2 Model's Training and Validation Accuracy-Loss Graphs with Adam Optimizer.

puting time. According to MobileNetV2, the training time increased partially as the depth of the proposed model increased. After removing blocks, adding dropouts, and freezing certain blocks, the trainable parameters have been reduced by approximately 4 times compared to the original MobileNetV2 and the weight of this model has been reduced to 11 MB. On the other hand, VGG19 and InceptionResNetV2 had the longest

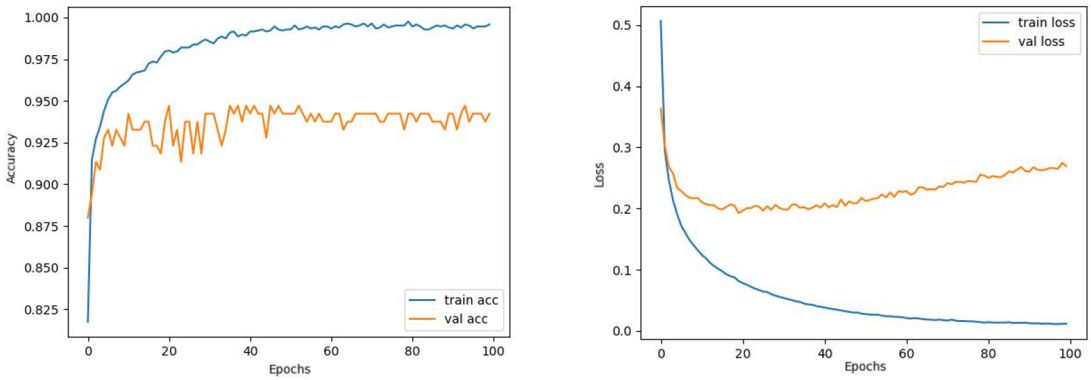

Figure 2.11: MobilenetV2 Model's Training and Validation Accuracy-Loss Graphs with Adamax Optimizer.

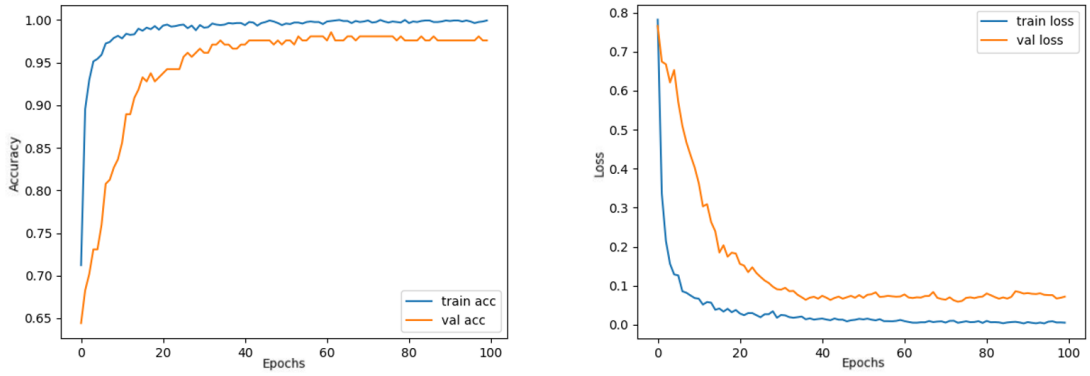

Figure 2.12: MobilenetV2 Modified Model's Training and Validation Accuracy-Loss Graphs with Adamax Optimizer.

Table 2.2: Comparison of Training Time and Model Weight Size for each Model

Model	**Training Time (min)**	**Model Weight Size**
InceptionV3	3:24	1.7 GB
VGG16	6:48	2.3 GB
VGG19	8:29	82 MB
ResNet50	7:13	1.4 GB
Xception	6:47	927 MB
InceptionResNetV2	8:30	264 MB
MobileNetV2	1:43	42 MB
MobileNetV2 Modified	1:25	11 MB

training time. It is worth noting that the InceptionV3 model required a considerable amount of training time, whereas the training time for VGG16 and Xception models was on average.

It is worth noting that previous studies have mainly focused on surface classification tasks by using both IMU and vision sensors. Most of them utilize conventional machine learning algorithms or deep learning networks, which attain reasonable results.

Table 2.3: Performance Results of Applied State-of-the-art Models and Proposed Model.

Model	Optimizer	Recall(%)	Precision(%)	F1-Score(%)	Train Acc.(%)	Train Loss(%)	Test Acc.(%)
InceptionV3	SGD	95	95	95	99.40	3.8	95.19
	Adam	95	95	95	99.64	0.5	95.67
	RMSprop	96	95	95	99.58	5.04	95.67
	Adadelta	92	93	92	95.74	14.45	92.79
	Adamax	95	95	95	99.52	2.56	95.67
	Adagrad	95	95	95	99.52	1.85	95.67
	Nadam	93	93	93	99.34	2.27	93.75
VGG16	SGD	86	87	86	88.77	35.42	87.02
	Adam	93	93	93	99.58	0.6	93.27
	RMSprop	92	92	92	99.46	0.7	91.83
	Adadelta	84	85	85	85.29	44.57	85.58
	Adamax	93	93	93	99.52	0.64	93.27
	Adagrad	91	91	91	93.63	22.95	91.93
	Nadam	95	95	95	99.40	0.59	95.19
VGG19	SGD	87	87	87	84.98	42.69	87.98
	Adam	91	91	90	99.58	2.46	91.35
	RMSprop	91	91	91	99.76	1.42	91.83
	Adadelta	52	56	49	59.76	93.38	57.69
	Adamax	94	94	94	96.76	13.47	94.23
	Adagrad	85	85	85	85.29	43.34	86.06
	Nadam	91	91	91	99.34	2.4	92.31
ResNet50	SGD	47	38	35	56.05	95.21	41.29
	Adam	56	60	57	61.08	90.50	56.73
	RMSprop	54	55	46	66.85	70.62	53.85
	Adadelta	55	50	48	56.16	91.66	57.69
	Adamax	54	55	52	72.07	61.23	57.21
	Adagrad	55	53	50	61.26	80.94	57.21
	Nadam	54	59	54	65.53	70.84	58.17
Xception	SGD	94	95	94	98.80	5.58	94.71
	Adam	93	93	93	99.58	0.55	93.75
	RMSprop	93	93	93	99.58	0.56	93.27
	Adadelta	93	94	93	97.36	9.80	93.75
	Adamax	93	93	93	99.40	0.69	93.75
	Adagrad	93	93	93	99.46	2.25	93.27
	Nadam	33	14	20	41.92	109	41.35
InceptionResnetV2	SGD	88	90	89	88.89	37.97	89.42
	Adam	95	95	95	99.58	0.61	95.19
	RMSprop	92	92	92	99.70	0.62	92.79
	Adadelta	86	88	86	85.83	67.83	87.50
	Adamax	94	94	94	99.52	1.26	94.71
	Adagrad	88	90	89	90.57	27.74	89.42
	Nadam	94	94	94	99.40	0.65	94.71
MobilenetV2	SGD	91	92	91	89.01	36.55	91.83
	Adam	93	93	93	99.52	1.31	93.27
	RMSprop	95	95	95	99.46	1.62	95.19
	Adadelta	79	82	80	79.52	69.71	81.25
	Adamax	95	95	95	99.58	1.15	95.67
	Adagrad	93	93	93	91.29	28.62	93.27
	Nadam	95	95	95	99.52	1.39	95.19
MobileNetV2-Modified	SGD	95	96	95	87.45	38.80	95.67
	Adam	99	99	99	99.82	1.08	98.56
	RMSprop	99	99	99	100	0.002	99.04
	Adadelta	70	74	70	65.71	99.88	73.08
	Adamax	**100**	**99**	**100**	**99.94**	**0.53**	**99.52**
	Adagrad	97	97	97	93.09	21.95	97.12
	Nadam	99	99	99	100	0.01	98.56

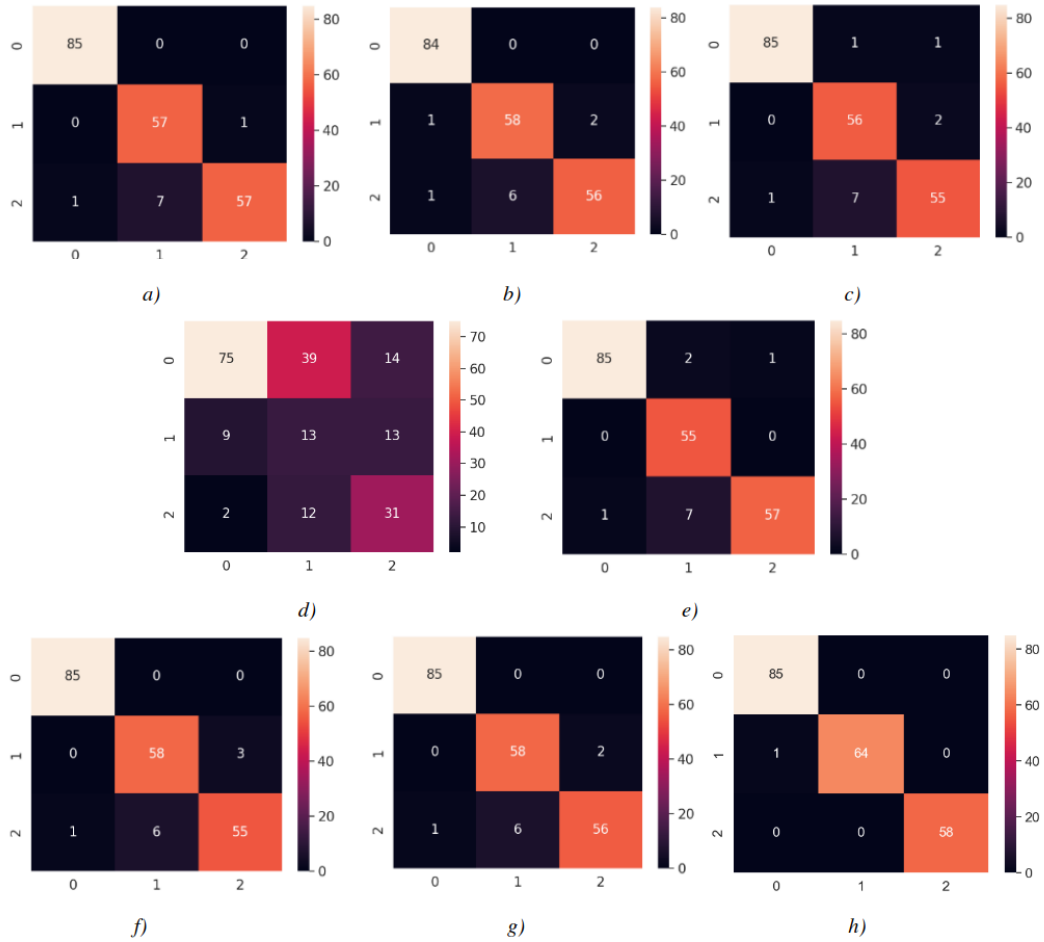

Figure 2.13: Confusion Matrices. a) InceptionV3 (Adam) b) VGG16 (Nadam) c) VGG19 (Adamax) d) ResNet50 (Adamax) e) Xception (SGD) f) InceptionResnetV2 (Adam) g) MobilenetV2 (Adamax) h) MobilenetV2 Modified (Adamax). Each model is associated with an optimizer performing best with the model.

Table 2.4: Comparison of the Proposed Model and Existing Studies

Research	**Sensor**	**Surface Type**	**Algorithm/Technique**	**Accuracy (%)**
[Kertész, 2016]	Ground Contact Force, Infrared, Accelerometer	Indoor	Random Forest (RF)	94
[Giguere and Dudek, 2011]	Tactile	Indoor/Outdoor	Artificial Neural Network(ANN)	94.6
[Lomio et al., 2019]	IMU	Indoor	XGBoost+Neural Network+ResNet	68.21
[Singh et al., 2023]	IMU	Indoor/Outdoor	Customized CNN Model	88
[Weiss et al., 2008]	Audio & Vision	Indoor/Outdoor	Support Vector Machine (SVM)	87.04
[Kurobe et al., 2021]	Audio & Vision	Indoor/Outdoor	ResNet50	80
[Guan et al., 2022]	IMU & Vision	Indoor/Outdoor	EfficientNet-B0 + IMU Denoising Module	98.37
Our study	**Vision**	**Indoor**	**MobileNetV2-Modified**	**99.52**

When Table 2.4 is examined, most of these studies were tested on indoor or outdoor surfaces by using more than one sensor. However, the accuracy remained within a

certain limit except for the work [Guan et al., 2022]. When they utilized only the vision sensor, the accuracy was around 90%, but when the IMU was added and the sensor fusion was supplied, it increased to around 98%. On the other hand, our study only used the vision sensor, and we facilitated its applicability in robotic applications by proposing a lightweight model called MobileNetV2-Modifed. Consequently, with these advantages, the proposed model achieved an accuracy of 99.52%, which outperformed other studies.

CHAPTER 3: SURFACE TYPE-BASED TASK ASSIGNMENT

This chapter is about determining the most efficient robot for processing carpets, tiles, and wood surfaces. This process requires the consideration of several factors. We performed a performance analysis of the Kobuki and Burger robots by considering several parameters such as the deviation angle, error to endpoint, and time. In this step, a multi-criteria decision making (MCDM) method was used to evaluate the performance of the robots based on the analysis. In addition, this method was used to determine which robot performed better on each surface. The application of the MCDM method helps to make decisions more effectively [Taherdoost and Madanchian, 2023a].

3.1 MULTI-CRITERIA DECISION-MAKING

Multi-criteria decision making (MCDM) techniques are mathematical and analytical approaches for resolving complicated decision issues incorporating a number of different criteria. Benjamin Franklin's publication on the concept of moral algebra is one of the first studies in the field of multicriteria decision making methods. Since the 1950s, studies of MCDM techniques have focused on their ability to be modeled mathematically to provide a framework which can aid in the structuring of decision issues [Taherdoost and Madanchian, 2023b]. MCDM includes several techniques that differ in various aspects. These methods differ mainly in how complex their algorithms are, how they weight the criteria, how they consider preferences in the evaluation, how they deal with ambiguous data, and finally how they aggregate the data [Bączkiewicz et al., 2021].

The Analytic Hierarchy Process (AHP) is one of many multi-criteria decision procedures that simplify the solution of complicated issues. According to the method's core principles, human reasoning and intuition are incorporated into it. The method supplies the capacity to rank the options by fusing logic and intuition. To assess the effectiveness of the alternatives and the relative significance coefficient of the criteria, pairwise comparisons are made. The significance coefficient in the AHP ranges from 1 to 10 [Strasser et al., 2002].

In this thesis, the performance analysis of the robots was performed using the AHP method. Therefore, by taking into account the sensor data like the IMU and the robot characteristics, a more accurate and comprehensive analysis was performed by applying the AHP.

3.2 TASK ASSIGNMENT USING MCDM

Criteria and their importance coefficients were determined for the Kobuki and Burger robots to perform performance scoring with the MCDM method. The following criteria were taken into account: angle of deviation, error to the end point, time, weight of the robot, and load capacity. We generated a score table by assigning importance coefficients based on our priorities using the MCDM method. The importance coefficients were graded on a scale of 1 to 10, where 1 denotes the least importance and 10 denotes the most importance.

It has identified exactly five criteria, but four of them have a negative impact. Therefore, the 'Load Capacity' value, which has a positive effect, is taken as minus (-). In other words, since the criteria that have a negative effect on the robot are entered as positive, the load capacity, which is the only positive criterion, will be entered and calculated as negative. For example, if we assume that the Burger robot makes a deviation of 7 degrees on the carpet surface and its importance coefficient is 10, it will be calculated as 70 points.

Table 3.1: Example of Multi Criteria Decision Making Scoring

	Criteria	**Significance Coefficient**	**Kobuki Values**	**Burger Values**	**Kobuki Score**	**Burger Score**
Surface Type	**Angle of Deviation (degree)**					
	Error to The End Point (cm)					
	Time (second)					
	Weight (kg)					
	Load Capacity (kg)					
				Total		

As can be seen in table 3.1, coefficients were set based on their significance for each criterion to analyze the performance of the robots. Later, the coefficients will be multiplied by the scores obtained from the Kobuki and Burger robots. Thus, the total score for both robots is obtained. Since we multiply the negative effects positively, the robot with a low total surface score performs better.

CHAPTER 4: EXPERIMENTS

In this chapter, we first introduce the robotic platforms used for the experiments. Second, we experimentally demonstrate the performance of the proposed model on the mobile robot under different indoor environments. Third, we cover the performance analysis of the Burger and Kobuki mobile robots. Finally, we experimentally demonstrate the task distribution according to the type of surface. In Algorithm 2, first, the

Algorithm 2 Pseudo Code of Surface Classification in Real Time

```
1: ImP=Load(Surface)
2: ImNorm=(ImP min(ImP))/(max(ImP)min(ImP))
3: Models ← { InceptionV3, VGG16, VGG19, Xception, ResNet, InceptionResNetV2,
   MobileNetV2, Proposed}
4: Optimizers ← {SGD, Adam, RMSProp, Adadelta, Adamax, Adagrad, Nadam}
5: for i=1 to |Models| do
6:    for j=1 to |Optimizers| do
7:       TrainedModel[i][j]=Models[i](ImNorm,Optimizers[j])
8:    end for
9: end for
10: for k=1 to TrainedModel[i][j] do
11:    ProposedModel ← BestWeights(TrainedModel[i][j])
12: end for
13: while Robot Moving do
14:    image ← CaptureImage every 5 seconds
15:    probs ← ProposedModel.Predict(image)
16:    surface_type ← max(probs)
17: end while
```

dataset with images of carpet, tile, and wood is loaded (line 1), and second, the images are normalized to reduce the computational cost (line 2). During training, eight different models are trained (line 3), consisting of seven different state-of-the-art models and one modified CNN-based model. Seven different optimizers (line 4) are used for each model. The models trained with different optimizers (line 10) are compared, and consequently, the model with the best performance (Proposed Model) is determined (line 11). Then, while the robot is moving (line 13), it captures an image from the camera every five seconds (line 14). It tests the captured images using the best model weight obtained in training and generates probabilities for three surface types (line 15). Finally, the algorithm returns the surface type with the highest probability (line 16).

4.1 EXPERIMENTAL PLATFORM

In this section, a detailed description of Kobuki and Burger, which are open-source differential-drive robots for indoor surface classification, is presented. The camera and IMU sensors used on the robots are explained. Besides this, the ROS platform used in the motion of the robots and the protocols used for the communication of the robots are mentioned. The experimental platforms of the Kobuki and Burger robots that we used during the tests are explained in detail in this section.

4.1.1 Kobuki

The Kobuki TurtleBot robot is a typical mobile robotic platform conceived for a variety of uses, including research, education, development, etc. Developed by the Yujin Robots which is South Korea based company, the Kobuki robot offers an efficient and reliable design. Having features such as long battery life, dependable odometry sensors, and power for external sensors and actuators makes it convenient for various indoor tasks. The view of the robot from different angles and the control panel is shown in Figure 4.1. Additionally, it supports the Robot Operating System (ROS) a strong framework that enables flexible development and the integration of different software modules. Thanks to its user-friendly interface and extensive documentation, the Kobuki mobile robot is a popular choice for robotics enthusiasts and professionals.

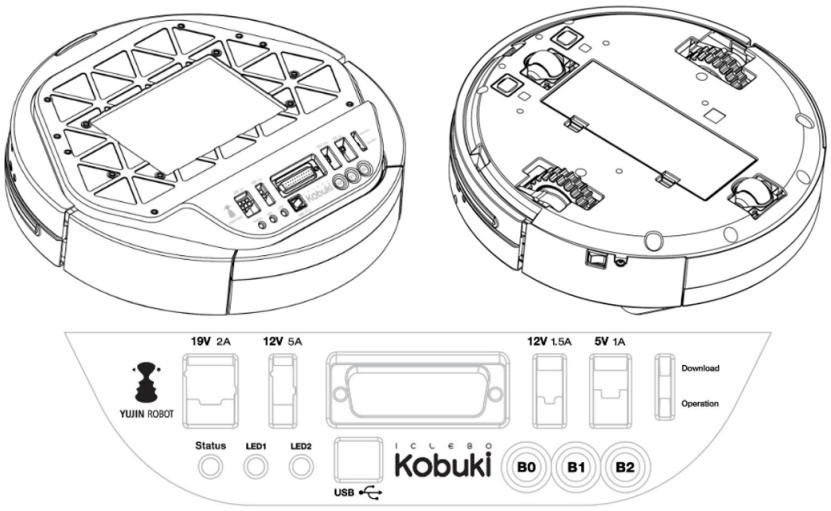

Figure 4.1: Top, Bottom and Control Panel Views of Kobuki Robot [Stonier, 2020]

The Kobuki robot utilized in this thesis is equipped with a laptop running the 64-bit Ubuntu 20.04 version of the Linux operating system and has an Intel i5 processor

with 8 GB of RAM. Figure 4.2 is an image taken from the Kobuki robot during the experiments.

Figure 4.2: Experimental Platform for Kobuki

Also, the camera and IMU sensors, whose descriptions will be given in the following sections, are included in the robot. The connection between the robot and the camera was established via a USB cable. After the connections, the laptop with the camera on it is placed on top of the robot. The Kobuki robot is composed of two differential drive wheels that can speed up to 0.7 m/s. The height of the robot, including the laptop with camera, is 270 mm, whereas the diameter is 350 mm. The weight of the robotic platform, including the components, is around 3 kg. More details about the specifications of the Kobuki robot can be found in Table 4.1.

Table 4.1: Features of Kobuki Robot

Item	**Features**
Maximum translational velocity	70 cm/s
Maximum rotational velocity	180 deg/s
Payload	5 kg (hard floor), 4 kg (carpet)
Threshold Climbing	12 mm or lower
Rug Climbing	12 mm or lower
Expected Operating Time	3 or 7 hours (small/large battery)
PC Connection	USB / RS232

4.1.2 Burger

TurtleBot Burgers are two-wheel differential drive mobile robots that are compact, dependable, and affordable. TurtleBot Burger3 is a model of the TurtleBot series and

is designed for research, education, and robotics projects. The view and dimensions of the robot from different angles are shown in Figure 4.3. Its objective is to drastically reduce the platform's size and cost without compromising its performance and quality. Each wheel of the robot is separately controlled. Additionally, this robot contains a Lidar sensor and a Raspberry Pi 3 Model B+ computer, which is useful for applying DL-based models.

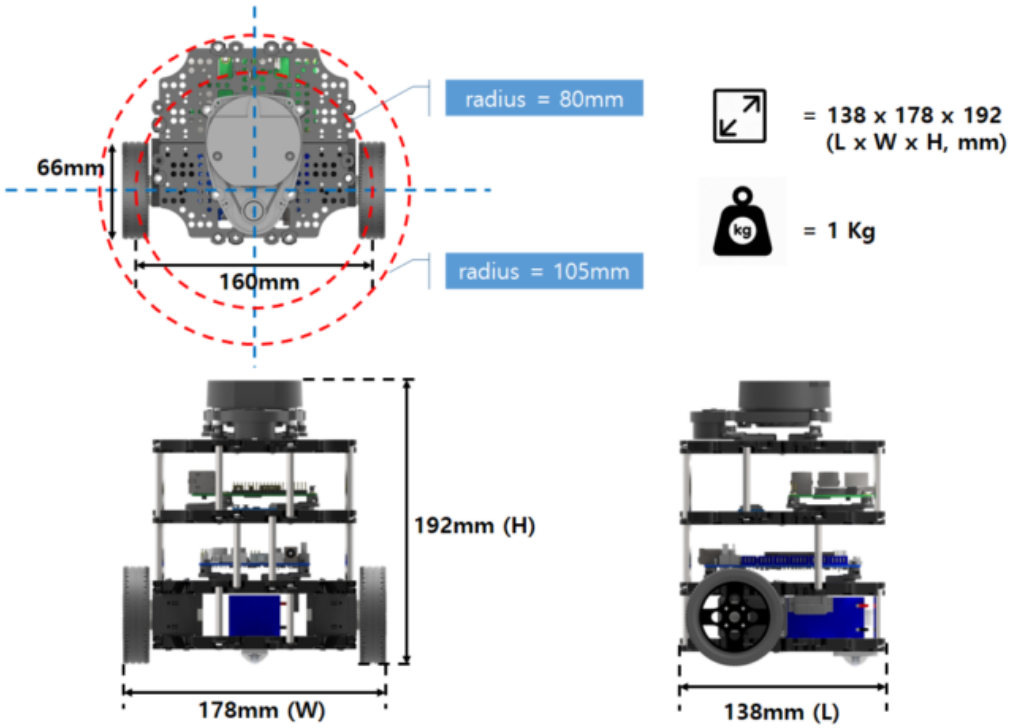

Figure 4.3: Front and Side Views of Burger Robot [Robotis, 2022]

The Burger robot used in this work consists of a Raspberry Pi 3 with a Cortex-A53 (ARMv8) CPU and 1GB of RAM running the Ubuntu 20.04 operating system. To perform the surface classification during the movement of the robot, the libraries TensorFlow (2.4.0) and OpenCV (4.7.0) were installed on the Raspberry Pi. The camera sensor is mounted on the robot platform, and the connection is provided via USB cable. Figure 4.4 is an image taken from the Burger robot during the experiments, showing the platform used.

The maximum translational velocity of the burger robots is 0.22 m/s, while the rotational speed is 2.84 rad/s, which corresponds to 162.72 deg/s. On average, the charge lasts up to 2.5 hours. It weighs around 1 kg, including the sensor, computer, and bat-

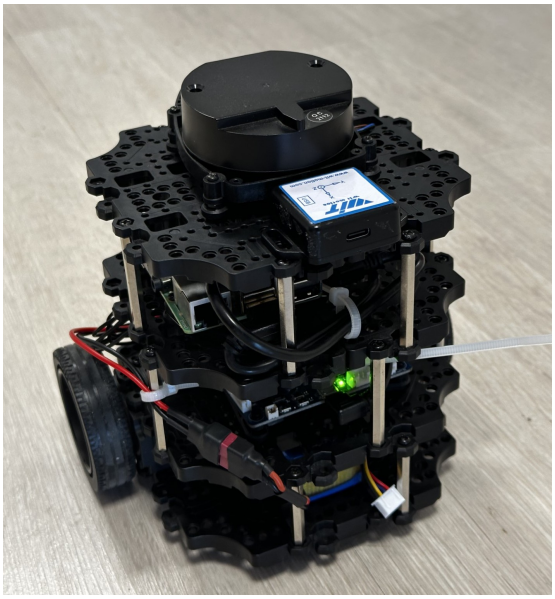

Figure 4.4: Experimental Platform for Burger.

tery. It has 138 mm of length, 178 mm of width, and 192 mm of height. More details about the specifications of the Burger robot can be seen in Table 4.2.

Table 4.2: Features of Turtlebot3 Burger Robot

Items	**Features**
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	162.72 deg/s
Maximum payload	15kg
Threshold of climbing	10 mm or lower
Expected operating time	2h 30m
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB

4.1.3 Camera and IMU Sensors

The Logitech C922 camera used in surface classification for both robots and the IMU (Inertial Measurement Unit) sensor, which is named WT901BLECL, used to detect deviation angles when evaluating the performance of robots during classification will be explained in this section. The camera used by the robot during surface classification and the IMU sensor, whose deviation angle we detected during the movement of the robot, can be seen in Figure 4.5.

During the experiments, we used a Logitech C922 webcam mounted on the robot for surface classification. This webcam offers a 78-degree field of view and is capable of capturing 1080p High Definition (HD) videos at a frame rate of 30 frames per sec-

Figure 4.5: Utilized Logitech C922 Camera and Wit Motion IMU Sensors

ond (fps). The selection of the Logitech C922 camera was based on its advantageous features, including high resolution, autofocus capabilities, and cost-effectiveness.

The IMU sensor is used for the motion and position tracking of objects. IMU sensors measure in three axes. It is used to detect the x-axis right-left, y-axis up-down, and z-axis rotation, that is, out-of-plane motion changes. In our thesis, this sensor shows how much the robot deviates from the target while moving on certain indoor surfaces. It is aimed that Kobuki and Burger robots will share tasks according to their performance on different indoor floors. For this, a 5-meter track was created on different indoor surfaces for both robots. While evaluating the performance, the total time the robot takes to reach the target is important, as well as the difference between the target point and the point it reaches, namely the error. In addition, another important point is how many degrees the robot deviates on the way to the target.

4.2 REAL TIME SURFACE CLASSIFICATION USING KOBUKI

To demonstrate the robustness of the proposed models in recognizing surface types, we conducted experiments in 11 different indoor environments. These environments include offices, rooms, and hallways. The surface types in each environment vary in terms of colors and patterns. We first transferred the weights of the proposed models (h5 file) to the laptop that was placed on top of the mobile robot. This made it suitable to perform surface classification on live streams while the robot is moving.

For the testing phase, 11 different scenarios were determined. Thus, we drove the mobile robot through passages where surfaces like wood-carpet-tile, carpet-tile, and

carpet-wood were present. During the movement of the mobile robot, an image was captured from a Logitech camera every 5 seconds and classified as carpet, tile, or wood. In total, 267 images were captured and tested. The proposed model correctly classified the 265 surface images, while only two images were classified incorrectly. These images belong to scenario 8 in Table 4.3. In Table 4.3 the number of correctly classified images for each scenario are shown. In scenarios where one or two surface types do not exist, they are presented with a dash (-).

Table 4.3: Classification results of the proposed model implemented on a mobile robot

Scenario	**Environment**	**Carpet**	**Tiles**	**Wood**
1	Hallway	14	12	-
2		-	15	-
3	Office 1	11	-	17
4	Office 2	9	-	13
5	Home 1	6	8	4
6		10	4	6
7		14	10	7
8	Home 2	10	8	10
9	Home 3	19	5	6
10		15	-	9
11		10	8	7

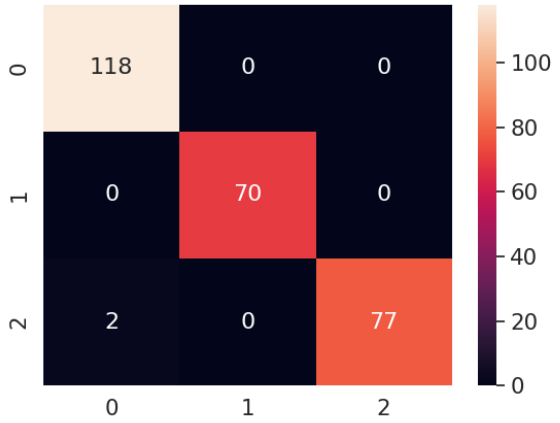

Figure 4.6: Confusion Matrix of Proposed Model for All the Scenarios.

The confusion matrix of the proposed model for all the scenarios is presented in Figure 4.6. It is clear that the proposed model correctly predicts all of the images except for two images. In scenario 8, the proposed method classified the carpet surface type as wood. To investigate the source of the misclassification, we repeated the experiment for scenario 8 by moving the robot over the same area. However, this time, it classified all surface types correctly. After analyzing the images from the first experiment and the

repeated experiment in scenario 8, we concluded that the misclassification was due to blurriness. Some of the correctly classified surface images and all misclassified images can be seen in Figures 4.7 and 4.8, respectively.

Figure 4.7: Samples of Correctly Classified Surface Types.

Figure 4.8: Blurry Images of Carpet Misclassified by the Proposed Model

In general, our model achieved an accuracy of 99.25% by correctly classifying 265 out of 267 surface images. Achieving high accuracy even in various indoor environments demonstrates the robustness of the proposed model.

4.3 MOTION PERFORMANCE ANALYSIS OF THE ROBOTS ON SURFACES

The performances of Burger and Kobuki robots were examined on carpet, tiles, and wood surfaces. We defined start and end points with different lengths on each surface. The performance was analyzed in terms of angular deviation, the error rate in centimeters to the destination point, and the time to reach the endpoint. Once the surface platform was ready, we moved both robots over to the same scenario. The angular

deviation was measured using the WT901BLECL IMU sensor, the error rate to the endpoint was determined using a digital laser meter, and the time was estimated via software. While the IMU Sensor can collect data for 9 axes, we only collected data from 3 axes to detect their angular deviations in X, Y, and Z. The X-axis represents forward motion, the Y-axis represents leftward motion, and the Z-axis represents upward motion. By performing a performance analysis of the robots, we determined which surface and which robot performed better. Therefore, based on the obtained data, tasks will be assigned to the robots.

To evaluate the performance of Burger and Kobuki robots, we designed tracks of 5 meters for each surface. The robots were tested at various velocities. While Kobuki was tested at 0.1, 0.2, and 0.3 m/s velocities, Burger was tested at 0.1, 0.15, and 0.2 due to its limited translational velocity. Figures 4.9, 4.10, and 4.11 show the test results for carpet, tile, and wood surfaces, using the same IMU sensor at a velocity of 0.2 m/s on a 5-meter track.

The resulting deviation angle of Burger and Kobuki robots on the carpet surface is demonstrated in Figure 4.9. The performance of the Kobuki robot on carpet surfaces is good, as can be seen in Figure 4.9. The carpet surface has challenging characteristics, such as soft and rough forms. Compared to the Burger, the Kobuki robot has a larger diameter and wheel size, which allows for better performance on the carpet despite its difficult characteristics.

On the other hand, when analyzing the movement of the Burger robot, it was observed that the rough, soft, and hollow structure of the carpet caused deviations during its movement. For example, the ball caster of the robot got stuck on the carpet yarns and caused deviations.

In the experiments conducted on the carpet, it was observed that the Kobuki robot moves stably along the track, with a deviation angle below 1°. While the burger robot remained stable until the 10th second, a sudden 13° deviation occurred. The reason for this is the change in texture induced by the carpet pattern.

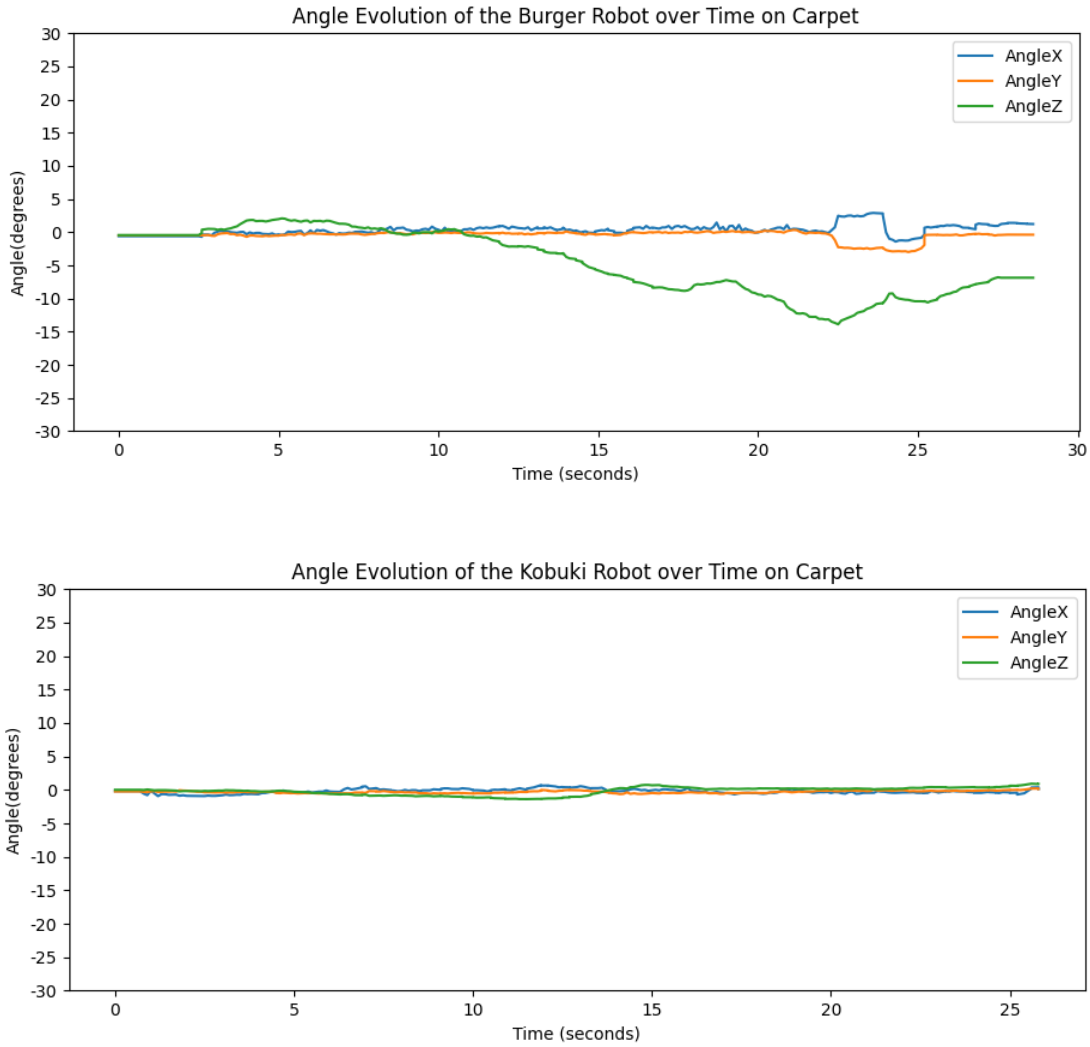

Figure 4.9: Angle Evolution of the Burger and Kobuki Robots on Carpet Surface

When examining the experiments of the Burger robot on tiles, it is evident that there are differences in the height of the tiles and joint gaps, causing the ball caster to stumble. Consequently, wobbling and sudden angular deviations occurred. In addition, tiles usually have a smooth and slippery surface, which may cause the wheels or moving parts of the robot to slip when there is insufficient friction. This can prevent the robot from moving at the desired speed and direction. Since the wheels of Kobuki are large, it is not seen that it makes sudden deflection movements between the joints like Burger. Despite this, 10 degrees of deflection of the robot were observed. This is because the friction coefficient of the floor is low, making it difficult to handle.

Although wooden surfaces may appear similar, their characteristics can vary significantly. The surface in the scenario we used for the experiment was found to be inclined

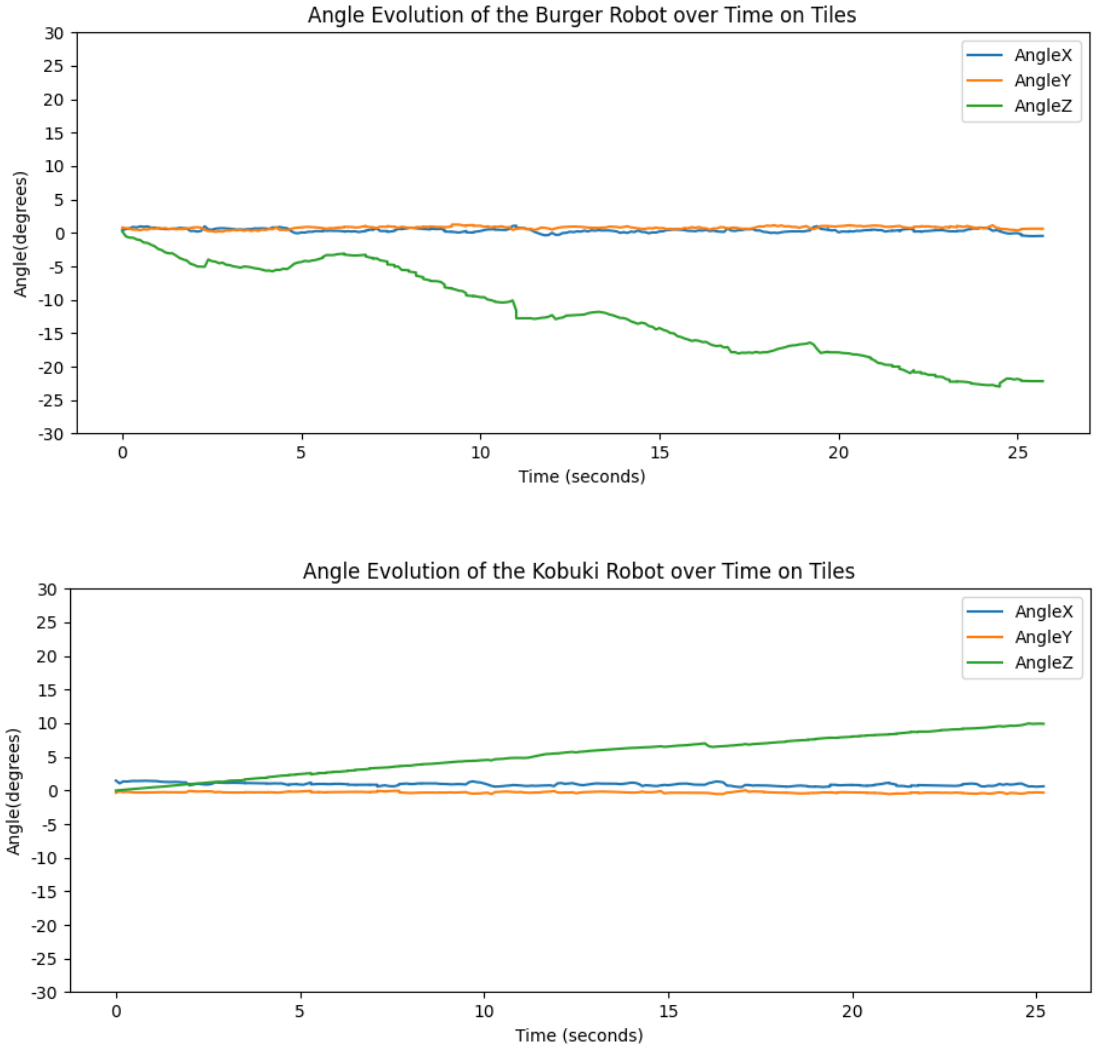

Figure 4.10: Angle Evolution of the Burger and Kobuki Robots on Tiles Surface

when measured with a spirit level. At the same time, the fact that the joints of the wood are not flat and dirty was among the factors that triggered the deviation in both the Kobuki and Burger robots.

The performance analysis of the robots in terms of deviation angle, error value to target, and time for carpet, tiles, and wood surfaces on a 5-meter track is depicted in Table 4.4. Each robot was tested with different velocities.

In Table 4.4, when the performance evaluation of the Kobuki and Burger robots on each surface is compared, it differs considerably. The Kobuki robot exhibits a higher degree of deflection on the carpet surface compared to the Burger robot, mainly due to getting caught in the yarns. Furthermore, when we consider the Burger robot in particular, the

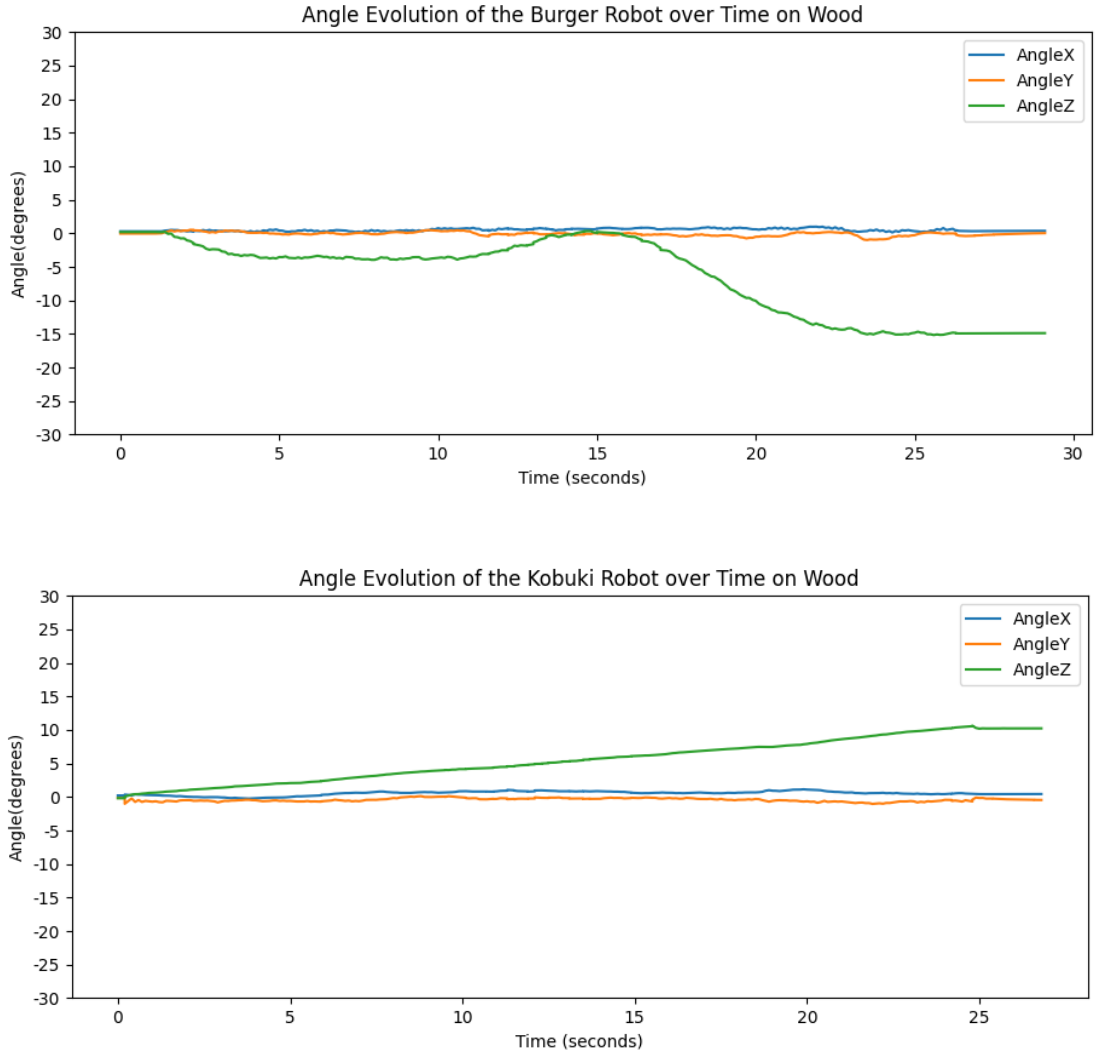

Figure 4.11: Angle Evolution of the Burger and Kobuki Robots on Wood Surface

margin of error when moving towards the target is higher on the carpet compared to other surfaces. Although both robots were affected due to the slippery and low friction coefficient of the tile, the Burger experienced additional negative effects due to the joint gaps, particularly impacting its ball caster. However, when we tested the Burger robot on tiles with not very deep joint gaps, it showed a significant improvement. The Burger robot achieved a minimal error value compared to the Kobuki robot on the wooden surface, but it exhibited an additional 2 ° of deviation. In general, the Burger performed better at lower speeds, given its maximum velocity of 0.22 m/s. On the other hand, the Kobuki robot was more stable in terms of deviation angle, although the margin of error was higher than the Burger.

The robot with the lowest score was considered the most effective. The results obtained

Table 4.4: Test Results for Performance Evaluation before Task Assignment in Kobuki and Burger Robots

Robots	**Surface Type**	**Velocity (m/s)**	**Deviation Angle (degree)**	**Error Value to Target (cm)**	**Time (seconds)**
Burger	Carpet	0.1	4.50	58	50.10
		0.15	4.85	52	33.40
		0.2	6.87	63	25.55
	Tiles	0.1	1.34	4	50.10
		0.15	29.12	26	33.40
		0.2	22.17	27	25.10
	Tiles without joint gaps	0.1	4.48	1	50.10
		0.15	15.93	10	33.40
		0.2	13.02	17	25.10
	Wood	0.1	2.3	1	50.10
		0.15	5.40	2	33.40
		0.2	12.19	11	25.10
Kobuki	Carpet	0.1	0.08	34	50.10
		0.2	0.88	39	25.10
		0.3	1.43	51	16.70
	Tiles	0.1	10.29	19	50.10
		0.2	9.90	30	25.10
		0.3	9.52	48	16.70
	Wood	0.1	7.77	18	50.10
		0.2	10.33	37	25.10
		0.3	10.19	44	16.70

according to this evaluation method are presented in Tables 4.5 and 4.6. When Table 4.5 is examined, it is evident that the Kobuki robot performs better on carpet and tile surfaces at a velocity of 0.2 m/s. On the other hand, the Burger robot shows more effective performance on the wood surface. In Table 4.6, the Burger robot is ahead in terms of scoring for tiles and wood. Since the maximum speed of the Burger robot was 0.22 m/s, it performed better at a velocity of 0.1 m/s in terms of its properties. Although the Kobuki robot had difficulties on the carpet, the Burger robot faced more challenges due to its wheels.

Table 4.5: Scoring of Mobile Robots with 0.2 Velocity for each Surface.

	Criteria	Significance Coefficient	Kobuki Values	Burger Values	Kobuki Score	Burger Score
Carpet	**Angle of Deviation (degree)**	10	0.88	6.87	8.8	68,7
	Error to The End Point (cm)	9	39	63	351	567
	Time (second)	4	25.1	25.55	100.4	102,2
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	4	15	20	75
				Total	**470.2**	**674.9**
Tiles	**Criteria**	**Significance Coefficient**	**Kobuki Values**	**Burger Values**	**Kobuki Score**	**Burger Score**
	Angle of Deviation (degree)	10	9.9	22.17	99	221.7
	Error to The End Point (cm)	9	30	27	270	243
	Time (second)	4	25.1	25.1	100.4	100.4
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	5	15	25	75
				Total	**474.4**	**502.1**
Wood	**Criteria**	**Significance Coefficient**	**Kobuki Values**	**Burger Values**	**Kobuki Score**	**Burger Score**
	Angle of Deviation (degree)	10	10.33	12.19	103.3	121.9
	Error to The End Point (cm)	9	37	11	333	99
	Time (second)	4	25.1	25.1	100.4	100.4
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	5	15	25	75
				Total	**541.7**	**258.3**

Table 4.6: Scoring of Mobile Robots with 0.1 Velocity for each Surface.

	Criteria	Significance Coefficient	Kobuki Values	Burger Values	Kobuki Score	Burger Score
Carpet	**Angle of Deviation (degree)**	10	0.08	4.50	0.8	45
	Error to The End Point (cm)	9	34	58	306	522
	Time (second)	4	50.1	50.1	200.4	200.4
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	4	15	20	75
				Total	**517.2**	**704.4**
Tile	**Criteria**	**Significance Coefficient**	**Kobuki Values**	**Burger Values**	**Kobuki Score**	**Burger Score**
	Angle of Deviation (degree)	10	10.29	1.34	102.9	13.4
	Error to The End Point (cm)	9	19	4	171	36
	Time (second)	4	50.1	50.1	200.4	200.4
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	5	15	25	75
				Total	**479.3**	**186.8**
Wood	**Criteria**	**Significance Coefficient**	**Kobuki Values**	**Burger Values**	**Kobuki Score**	**Burger Score**
	Angle of Deviation (degree)	10	7.77	2,3	77.7	23
	Error to The End Point (cm)	9	18	1	162	9
	Time (second)	4	50.1	50.1	200.4	200.4
	Weight (kg)	6	5	2	30	12
	Load Capacity (kg)	5	5	15	25	75
				Total	**445.1**	**169.4**

4.4 MULTI-ROBOT EXPERIMENTS

After training CNN-based models and the proposed model, it was determined that the proposed model performed better. Therefore, in the experiments of this section, we used the proposed model weights, just like in the prior experiments. To assign tasks to the robots according to the surface type, the performance analysis of the robots was done by considering three different parameters, as mentioned above. Subsequently, the MCDM method was applied to make the analysis more efficient. Thus, it was determined which robot performed better on which ground. In this section, the motion experiments of the robots will be carried out according to the surface.

To perform the experiments, we utilized Kobuki and Burger robots. According to the scenario in the experiments, communication is provided by three computers. Kobuki was determined to be the leader, and Burger Robot was determined to be a member among these computers. Besides, there is a ground computer where we get the classification results as strings. The communication between Kobuki and Burger is provided via Transmission Control Protocol (TCP). The representation of the communication phase between robots is demonstrated in Figure 4.12.

The Kobuki leader computer consists of three nodes. These are classification, communication, and motion nodes, respectively. First, the classification node captures a real-time image from the camera and classifies the surface type by utilizing the proposed model weights. Then the classification result message via $/surface\_type$ topic is transferred to the Motion node, which moves the Kobuki robot, as well as the Leader node, which communicates with the Burger and ground computers. In other words, the Classification node is the publisher of the topic named $/surface\_type$, and the Motion and Leader nodes are subscribers. The Kobuki robot has been more successful on carpet and tiles surfaces when performance analysis is made based on the Multi Criteria Decision Making method. Therefore, for the Kobuki robot, when the Motion Node receives carpet and tile messages, it will provide motion and perform its task. The leader communication node transmits the message of the surface type obtained from the classification node to the communication nodes of the member and ground computers. When the communication node of the member computer receives messages

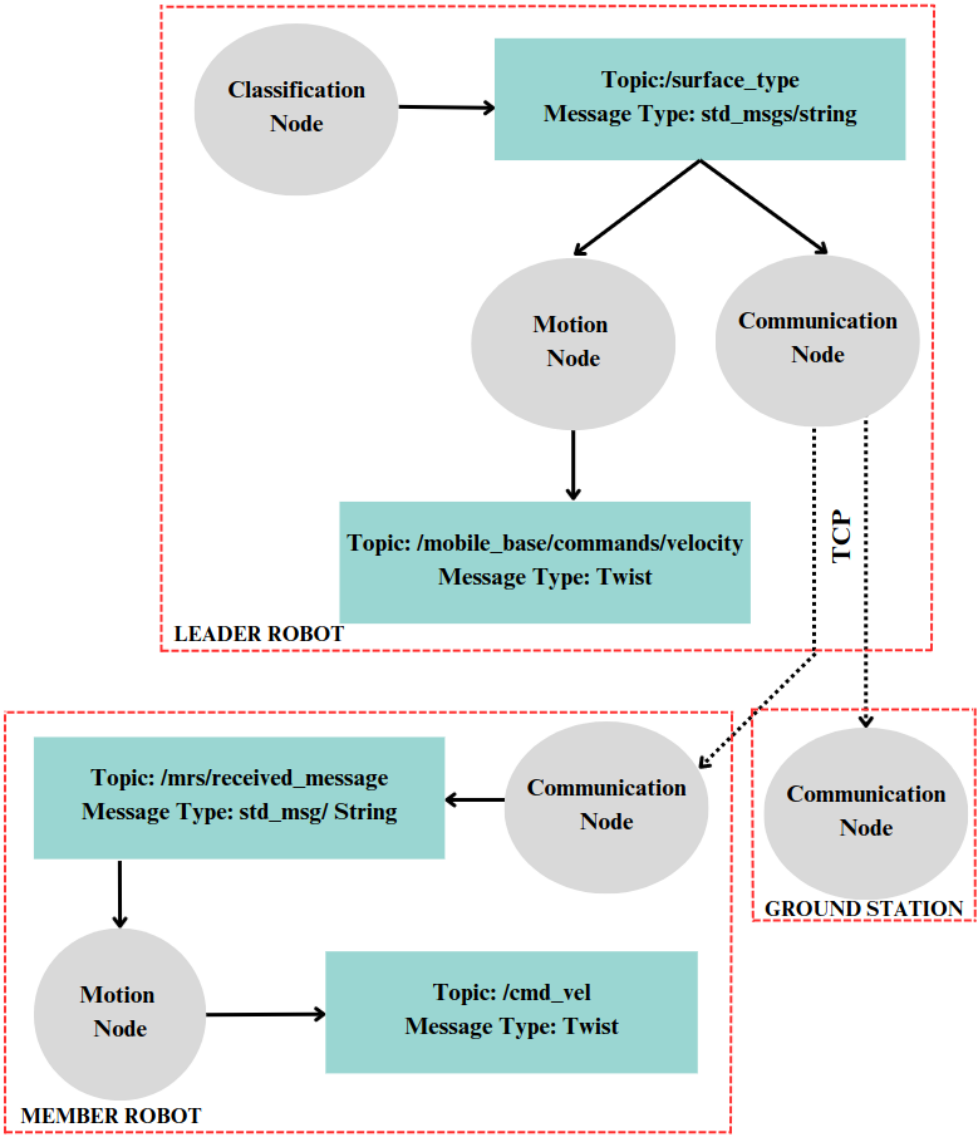

Figure 4.12: ROS Infrastructure of the Kobuki and Burger robots.

called carpet and tiles Burger robot will not act, when it receives a wood message, Burger will move and perform his task. The velocity in motion nodes for the Kobuki and Burger robots are published in Twist type. The Kobuki publishes velocity messages via the */mobile_base/commands/velocity* topic, while Burger via */cmd_vel*. As a result, when the leader robot classifies the type of surface as carpet or tile, it will move itself; however, when it classifies the surface as wood, the Burger robot will move.

For the experiments, four different scenarios were conducted. Each scenario consists of different surface types. We tested the robots in kitchen, laboratory, and hallway environments with tile-wood, carpet-wood, just carpet, and just wood floors. In the

first experiment, we tested both robots on a floor with tile-wood, Kobuki was placed on tile, and Burger on wood. Kobuki moves itself during surface classification, resulting in tiles. When the classification result was wood, it stopped moving, and the Burger robot started moving. The second scenario was performed on a carpet-wood floor. Kobuki was placed on a carpet, and it moved itself around 1 meter during the prediction. When it passes the carpet floor and reaches the wood, the Burger robot starts moving. Later, we placed both robots on the carpet surface. The Kobuki robot moved to the end of the carpet, while the Burger robot was stable in its place. Further, on the wood surface, Burger was moving, while Kobuki was steady. Some samples taken during the experiments are demonstrated in Figure 4.13. When all scenarios are examined, it has been seen that the task assignment process to the robots according to the ground type is performed with high performance, and the robots adjust their movements according to the surface type.

Figure 4.13: Sample of both Robots on Carpet and Wood Surfaces.

CHAPTER 5: CONCLUSION

Towards the improvement of indoor mobile robots for surface classification tasks, we developed a lightweight MobileNetV2-modified model. We then analyzed the performance of pre-trained state-of-the-art DL based models, including InceptionV3, VGG16, VGG19, ResNet50, Xception, InceptionResnetV2, and MobileNetV2. Our proposed model's total parameters are reduced approximately four times compared to the original MobileNetV2 model. This parameter reduction makes the model well-suited for use in computationally limited systems. Moreover, we generated a unique dataset with various types of carpets, tiles, and wood collected from different indoor environments. Finally, we applied several optimizers, namely SGD, RMSProp, Adam, Adadelta, Adamax, Adagrad, and Nadam, to determine the most efficient model. The experiments show that the proposed MobileNetV2-modified model outperforms all applied state-of-the-art DL models and existing approaches in the literature. Besides this, the robustness of the proposed model is demonstrated by its high accuracy performance in real-time indoor scenarios when examined on the mobile robot. To evaluate the performance of the Kobuki and Burger robots on different surfaces, a performance analysis was carried out in terms of angular deviation, error to the desired point, and time. To make the analysis more efficient, we applied the MCDM method. Therefore, we determined which robot's performance was better on which surface. Following, we perform experiments with multi-robot task assignments according to the surface type. The task assignment experiments show that both robots performances were successful.

Future work includes enhancing the system by removing blurriness with the help of image processing filters and computer vision techniques. Additionally, the aim is to expand the applicability of this system in various indoor environments by adding new surface types to our dataset. Finally, by implementing mapping and localization methods, it is aimed to improve the dependability and effectiveness of mobile robots in performing their assigned tasks, even in complex environments.

REFERENCES

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Bączkiewicz et al., 2021] Bączkiewicz, A., Wątróbski, J., Kizielewicz, B., and Sałabun, W. (2021). Towards objectification of multi-criteria assessments: a comparative study on mcda methods. In 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), pages 417–425. IEEE.

[Bai et al., 2019] Bai, C., Guo, J., and Zheng, H. (2019). Three-dimensional vibration-based terrain classification for mobile robots. IEEE Access, 7:63485–63492.

[Bermudez et al., 2012] Bermudez, F. L. G., Julian, R. C., Haldane, D. W., Abbeel, P., and Fearing, R. S. (2012). Performance analysis and terrain classification for a legged robot over rough terrain. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 513–519. IEEE.

[Bosworth et al., 2016] Bosworth, W., Whitney, J., Kim, S., and Hogan, N. (2016). Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 3582–3589. IEEE.

[Chollet, 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1251–1258.

[Demirtas et al., 2023] Demirtas, A., Bayram, H., and Erdemir, G. (2023). Dataset for indoor surface classification, https://github.com/fieldroboticslab/dataset-for-indoor-surface-classification. Accessed: 2023-05-26.

[Diddeniya et al., 2019] Diddeniya, S., Wanniarachchi, W., De Silva, P., and Ganegoda, N. (2019). Efficient office assistant robot system: autonomous navigation and controlling based on ros. International Journal of Multidisciplinary Studies, 6:67–71.

[Dozat, 2016] Dozat, T. (2016). Incorporating nesterov momentum into adam. In 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Workshop Track Proceedings, pages 1–4.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7).

[Giguere and Dudek, 2011] Giguere, P. and Dudek, G. (2011). A simple tactile probe for surface identification by mobile robots. IEEE Transactions on Robotics, 27(3):534–544.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT press.

[Guan et al., 2022] Guan, T., Song, R., Ye, Z., and Zhang, L. (2022). Vinet: Visual and inertial-based terrain classification and adaptive navigation over unknown terrain. arXiv preprint arXiv:2209.07725.

[Guan et al., 2021] Guan, W. H., Melvern, C., Hou, F. T., Zaw, A. M., Khan, M. A., Aramugam, K., and Ramaswamy, M. (2021). D-bot: A food serving robot during pandemic situation. In 2021 IEEE International Conference on Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON), pages 22–25. IEEE.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

[Hinton et al., 2012] Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14(8):2.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in science & engineering, 9(03):90–95.

[Kertész, 2016] Kertész, C. (2016). Rigidity-based surface recognition for a domestic legged robot. IEEE Robotics and automation letters, 1(1):309–315.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[Kurobe et al., 2021] Kurobe, A., Nakajima, Y., Kitani, K., and Saito, H. (2021). Audio-visual self-supervised terrain type recognition for ground mobile platforms. IEEE Access, 9:29970–29979.

[Lomio et al., 2019] Lomio, F., Skenderi, E., Mohamadi, D., Collin, J., Ghabcheloo, R., and Huttunen, H. (2019). Surface type classification for autonomous robot indoor navigation. arXiv preprint arXiv:1905.00252.

[Mukkamala and Hein, 2017] Mukkamala, M. C. and Hein, M. (2017). Variants of rmsprop and adagrad with logarithmic regret bounds. In International conference on machine learning, pages 2545–2553. PMLR.

[Muthugala et al., 2020] Muthugala, M. V. J., Vengadesh, A., Wu, X., Elara, M. R., Iwase, M., Sun, L., and Hao, J. (2020). Expressing attention requirement of a floor cleaning robot through interactive lights. Automation in Construction, 110:103015.

[Niloy et al., 2021] Niloy, M. A., Shama, A., Chakrabortty, R. K., Ryan, M. J., Badal, F. R., Tasneem, Z., Ahamed, M. H., Moyeen, S. I., Das, S. K., Ali, M. F., et al. (2021). Critical design and control issues of indoor autonomous mobile robots: A review. IEEE Access, 9:35338–35370.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. the Journal of machine Learning research, 12:2825–2830.

[Rajendran et al., 2020] Rajendran, G. B., Kumarasamy, U. M., Zarro, C., Divakarachari, P. B., and Ullo, S. L. (2020). Land-use and land-cover classification using a human group-based particle swarm optimization algorithm with an lstm classifier on hybrid pre-processing remote-sensing images. Remote Sensing, 12(24):4135.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. The annals of mathematical statistics, pages 400–407.

[Robotis, 2022] Robotis (2022). Turtlebot3 Burger Manual. https://emanual.robotis.com/docs/en/platform/turtlebot3/features/specifications. Accessed: 02.02.2023.

[Rubio et al., 2019] Rubio, F., Valero, F., and Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems, 16(2):1729881419839596.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115:211–252.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[Singh et al., 2023] Singh, S., Sajwan, M., Singh, G., Dixit, A. K., and Mehta, A. (2023). Efficient surface detection for assisting collaborative robots. Robotics and Autonomous Systems, 161:104339.

[Stefek et al., 2020] Stefek, A., Van Pham, T., Krivanek, V., and Pham, K. L. (2020). Energy comparison of controllers used for a differential drive wheeled mobile robot. IEEE Access, 8:170915–170927.

[Stonier, 2020] Stonier, D. (2020). Kobuki Documentation Release 1.0.0. https://kobuki.readthedocs.io/$_{/}$*downloads/en/devel/pdf/*. *Accessed* : 02.02.2023.

[Strasser et al., 2002] Strasser, S. E., Ozgur, C., and Schroeder, D. L. (2002). Selecting a business college major: An analysis of criteria and choice using the analytical hierarchy process. American Journal of Business, 17(2):47–56.

[Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence, volume 31.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9.

[Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826.

[Taherdoost and Madanchian, 2023a] Taherdoost, H. and Madanchian, M. (2023a). Multi-Criteria Decision Making (MCDM) Methods and Concepts. Encyclopedia, 3(1):77–87.

[Taherdoost and Madanchian, 2023b] Taherdoost, H. and Madanchian, M. (2023b). Multi-criteria decision making (mcdm) methods and concepts. Encyclopedia, 3(1):77–87.

[Tick et al., 2012] Tick, D., Rahman, T., Busso, C., and Gans, N. (2012). Indoor robotic terrain classification via angular velocity based hierarchical classifier selection. In 2012 IEEE International Conference on Robotics and Automation, pages 3594–3600. IEEE.

[Waskom et al., 2017] Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., Augspurger, T., Halchenko, Y., Cole, J. B., Warmenhoven, J., de Ruiter, J., Pye, C., Hoyer, S., Vanderplas, J., Villalba, S., Kunter, G., Quintero, E., Bachant, P., Martin, M., Meyer, K., Miles, A., Ram, Y., Yarkoni, T., Williams, M. L., Evans, C., Fitzgerald, C., Brian, Fonnesbeck, C., Lee, A., and Qalieh, A. (2017). mwaskom/seaborn: v0.8.1 (september 2017).

[Weiss et al., 2008] Weiss, C., Tamimi, H., and Zell, A. (2008). A combination of vision- and vibration-based terrain classification. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2204–2209. IEEE.

[Xue et al., 2022] Xue, F., Hu, L., Yao, C., Liu, Z., Zhu, Z., and Jia, Z. (2022). Sound-based terrain classification for multi-modal wheel-leg robots. In 2022 International Conference on Advanced Robotics and Mechatronics (ICARM), pages 174–179. IEEE.

[Yasuda et al., 2020] Yasuda, Y. D., Martins, L. E. G., and Cappabianco, F. A. (2020). Autonomous visual navigation for mobile robots: A systematic literature review. ACM Computing Surveys (CSUR), 53(1):1–34.

[Zeiler, 2012] Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.