Derin Sinir Ağları İçin Mutasyon Tabanlı Test Kütüphanesi Geliştirilmesi

Gökhan Çetiner

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Nisan 2025

Development of a Mutation-Based Test Library for Deep Neural Networks

Gökhan Çetiner

MASTER OF SCIENCE THESIS

Department of Computer Engineering

April 2025

Derin Sinir Ağları İçin Mutasyon Tabanlı Test Kütüphanesi Geliştirilmesi

Gökhan Çetiner

Eskişehir Osmangazi Üniversitesi

Fen Bilimleri Enstitüsü

Lisansüstü Eğitim ve Öğretim Yönetmeliği Uyarınca

Bilgisayar Mühendisliği Anabilim Dalı

Yapay Zekâ Bilim Dalında

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

Danışman: Doç. Uğur YAYAN
İkinci Danışman: Prof. Dr. Ahmet YAZICI

Bu tez, TUBİTAK 22AG040 nolu "Sürdürülebilir Kentler için İleri Teknolojiler Platformu (SÜİT)" projesi tarafından desteklenmiştir.

Nisan 2025

ONAY

Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans öğrencisi **Gökhan Çetiner**'in YÜKSEK LİSANS tezi olarak hazırladığı "Derin Sinir Ağları İçin Mutasyon Tabanlı Test Kütüphanesi Geliştirilmesi" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek oybirliği ile kabul edilmiştir.

Danışman : Doç. Uğur YAYAN

İkinci Danışman : Prof. Dr. Ahmet YAZICI

Yüksek Lisans Tez Savunma Jürisi:

Üye: Doç. Dr. Uğur YAYAN

Üye: Prof. Dr. Kemal ÖZKAN

Üye: Dr. Öğr. Üyesi Emre GÜNGÖR

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……..…….. tarih ve ……..……. sayılı kararıyla onaylanmıştır.

Prof. Dr. ....

Enstitü Müdürü

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Doç. Dr. Uğur YAYAN ve Prof. Dr. Ahmet YAZICI danışmanlığında hazırlamış olduğum "Derin Sinir Ağları İçin Mutasyon Tabanlı Test Kütüphanesi Geliştirilmesi" başlıklı Yüksek Lisans tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallarına uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 21/04/2025

Gökhan ÇETİNER

İmza

ÖZET

Derin Sinir Ağları (DNN), otonom araçlar ve üretken yapay zekâ sistemleri gibi birçok kritik alanda kullanılmaktadır. Otonom araçlar gibi kritik alanlarda kullanılan modellerin risklerden dolayı hatasız ve test edilmiş olarak kullanıma sunulmaları gerekmektedir. Özellikle kritik alanlarda kullanılan modeller için mutasyon testi hayati önem taşımaktadır. Mutasyon tabanlı test, DNN modellerinin karmaşık yapılarına mutasyonlar uygulanarak gerçekleştirilen, etkili bir test yöntemidir. Derin Mutasyon Modülü (DMM), DNN sistemlerinin test ihtiyacını karşılamak üzere geliştirilmiştir. DMM, mutasyon kütüphanesini aracılığıyla parametrelere özgü mutasyonlar ile DNN modellerinin mutasyon testine olanak tanır. Karmaşık DNN yapılarındaki hataların tespiti ve değerlendirilmesi, karmaşık ve devam eden bir çözüm gerektirmektedir. Bu tez kapsamında sunulan yaklaşım, mutasyonlar yoluyla modellerdeki hataları ve yapılandırma sorunlarını ortaya çıkarmaktadır. Hayatta kalan mutantlar üzerinden yapılan analizler, geliştiricilere modelin başarımını artırmaya yönelik önemli geri bildirimler sunmaktadır. Bu yöntem aracılığıyla test edilen modelin emniyeti mutasyon testleriyle ile analiz edilmektedir. Yapılan testler, elektrikli araç rotalaması için Pekiştirmeli Öğrenme (Reinforcement Learning) modeli, Dönüştürücü (Transformer) modeli ve prognostik tahminler için Uzun Kısa Süreli Bellek (Long Short-Term Memory) modeli dâhil olmak üzere üç farklı model türü için mutasyon testine odaklanmaktadır. LSTM modeline uygulanan mutasyon testleri sonucunda, sırasıyla 5, 10, 20 ve 30 dönem (epoch) için %96,61, %91,02, %71,19 ve %68,77 oranlarında mutasyon skorları kaydedilmiştir. RL modeli testlerinde üç farklı veri seti için mutasyon skorları 78,23%, 78,20% ve 70,83% olarak elde edilmiştir. Dönüştürücü modeli testlerinde sırasıyla 5,10 ve 20 dönem için %75,87, %76,36 ve %74,93 başarım skoru alınan üç model testinden 66,8%, 73,3% ve 64,9% mutasyon skoru elde edilmiştir. DMM, testlerde mutantları “hayatta kaldı” (survived) ve “öldürüldü” (killed) olarak sınıflandırarak yalnızca geliştirme sürecine destek sağlamakla kalmaz, aynı zamanda geliştirilen modellerin emniyetli ve uygulamaya hazır olduklarını doğrulama sürecinde de kritik bir rol oynar.

Anahtar Kelimeler: derin sinir ağları, uzun kısa süreli bellek, makine öğrenimi, mutasyona dayalı test, pekiştirmeli öğrenme, dönüştürücü mimarisi

SUMMARY

Deep Neural Networks (DNNs) are utilized in various critical domains, including autonomous vehicles and generative artificial intelligence systems. Due to the risks associated with these domains, models deployed in safety-critical applications such as autonomous vehicles must be error-free and thoroughly tested. Mutation testing is of vital importance for models used in such critical contexts. Mutation-based testing is an effective method in which artificial mutations are applied to the complex structures of DNNs. The Deep Mutation Module (DMM) has been developed to address the testing needs of DNN systems. Through its mutation library, DMM enables mutation testing by applying parameter-specific mutations to DNN models. Detecting and evaluating faults within complex DNN architectures requires an ongoing and systematic solution. The approach presented in this thesis reveals faults and configuration issues in models through targeted mutations. Analyses performed on the surviving mutants provide developers with valuable feedback for improving model performance. This method also allows for the analysis of the safety of the tested model using mutation tests. The conducted experiments focus on three different model types: a Reinforcement Learning (RL) model for electric vehicle routing, a Transformer model, and a Long Short-Term Memory (LSTM) model for prognostic predictions. As a result of mutation testing on the LSTM model, mutation scores of 96.61%, 91.02%, 71.19%, and 68.77% were recorded for 5, 10, 20, and 30 epochs, respectively. For the RL model, mutation scores of 78.23%, 78.20%, and 70.83% were obtained across three different datasets. In the Transformer model tests, performance scores of 75.87%, 76.36%, and 74.93% were recorded for 5, 10, and 20 epochs respectively, and the corresponding mutation scores were 66.8%, 73.3%, and 64.9%. By classifying mutants as "survived" and "killed", DMM not only supports the model development process but also plays a critical role in verifying that the developed models are safe and ready for deployment.

Keywords: deep neural networks, long short-term memory, machine learning, mutation-based testing, reinforcement learning, transformers

TEŞEKKÜR

Bu tez, TUBİTAK 22AG040 nolu “Sürdürülebilir Kentler için İleri Teknolojiler Platformu (SÜİT)” projesi tarafından desteklenmiştir. Projeye verdiği destekten ötürü TÜBİTAK’a teşekkürlerimizi sunarız.

Akademik çalışmam süresince çalışmalarıma desteğini esirgemeyen ve gösterdiği anlayışla her adımda katkı sağlayan Doç. Dr. Uğur YAYAN ve Prof. Dr. Ahmet YAZICI hocama rehberliklerinden dolayı teşekkürlerimi sunuyorum.

İÇİNDEKİLER

Sayfa

ÖZET..... vi
SUMMARY..... vii
TEŞEKKÜR..... viii
İÇİNDEKİLER..... ix
ŞEKİLLER DİZİNİ..... x
ÇİZELGELER DİZİNİ..... xi
SİMGELER VE KISALTMALAR DİZİNİ..... xiii
1. GİRİŞ VE AMAÇ..... 1
2. LİTERATÜR ARAŞTIRMASI..... 6
2.1 Derin Sinir Ağları İçin Test Yaklaşımları..... 6
2.2 Derin Sinir Ağları Modellerinde Mutasyon Testi……..... 7
2.3 Derin Sinir Ağları Modellerinde Test Araçları..... 8
3. MATERYAL VE YÖNTEM..... 11
3.1. Test Ortamı..... 11
3.2. Test Proje Ortamı..... 13
3.3. Derin Mutasyon Modülü 14
3.3.1 Önerilen derin mutasyon modülü yöntemi..... 15
3.3.2 Derin mutasyon modülü tarama işlemi 22
3.3.3 Derin mutasyon modülü test yürütme işlemi…..... 24
3.4. Test Ön Hazırlıklar 25
4. BULGULAR VE TARTIŞMA..... 28
4.1. Elektrikli Araç Filoları İçin Otonom Yönetim Sistemi LSTM Modeli …….. 31

İÇİNDEKİLER

Sayfa

4.2. Elektrikli Araç Filoları için Otonom Yönetim Sistemi İçin Pekiştirmeli Öğrenme (RL) ile Güzergâh Optimizasyonu..... 36

4.3. Dönüştürücü Tabanlı Model Testi..... 36

4.4. Tartışma……..... 39

5.SONUÇ VE ÖNERİLER……..... 42

KAYNAKLAR DİZİNİ..... 44

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

3.1 Proje Ortamı……..……..……12

3.2 Derin Mutasyon Modülü Akış Diyagramı……..……17

3.3 Derin Mutasyon Modülü mutasyon algoritması……..……20

3.4 Derin Mutasyon Modülü Kullanıcı Arayüzü……..…….23

3.5 Mutasyon Planı……..…….25

4.1 LSTM Mutasyon testi sonuçları……..……..…….30

4.2 LSTM dönem sayısına göre mutasyon testi sonuçları ……..……..……30

4.3 LSTM mutasyon testi skorları…..……..…….……31

4.4 Rastgele RL mutasyon mesti sonuçları ……..……34

4.5 Kümelenmiş RL mutasyon testi sonuçları …….……34

4.6. Rastgele Kümelenmiş RL mutasyon testi sonuçları…….…….…….35

4.7.Veri setlerine göre mutasyon skoru…….…….……..……....35

4.8 Döneme göre mutasyon skorları …..……..37

4.9 Hedef modelin eşik değerleri ……...…….…….38

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

3.1 Model tipine göre değerlendirme koşulları ……..……..……..…….. 18

3.2 Modül kütüphanesi parametre kapsamları ….....…….. 22

3.3 RL modeli mutasyona uğramış parametreler…….....……... 26

3.4 LSTM modeli mutasyona uğramış parametreler…….....…… …… 27

3.5 Dönüştürücü modeli mutasyona uğramış parametreler……..... ……..…… 27

4.1 LSTM modeli mutasyon örneği.……..…….. 29

4.2 RL modeli mutasyon örneği.…….……..…….…….. 33

4.3 RL testleri veri seti ve epizot sayısına göre mutasyon skorları…….… 36

4.4 Dönüştürücü modeli mutasyon örneği …….…..……... 37

4.5 Sınıflandırmaya Göre mutasyon testi sonuçları ……. 37

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
DNN	Derin Sinir Ağları (Deep Neural Networks)
ML	Makine Öğrenmesi (Machine Learning)
AI	Yapay Zekâ (Artificial Intelligence)
LSTM	Uzun Kısa Süreli Bellek (Long Short-Term Memory)
CNN	Evrişimli Sinir Ağları (Convolutional Neural Networks)
RL	Pekiştirmeli Öğrenme (Reinforcement Learning)
LLM	Büyük Dil Modelleri (Large Language Models)
IMFIT	Özelleştirilmiş mutasyon tabanlı yazılım hata enjeksiyon aracı (Tailored mutation-based software fault injection tool)
DMM	Derin Mutasyon Modülü (Deep Mutation Module)
SÜİT	Sürdürülebilir Kentler için İleri Teknolojiler Platformu

1. GİRİŞ VE AMAÇ

Derin Sinir Ağları (DNN) tabanlı Yapay Zekâ (AI) yöntemlerinin kullanımı artmaktadır (Sze vd., 2017). DNN'ler, AI sistemlerinin hızlı büyümesinde kritik bir rol oynamaktadır. Yapay zekâ, özellikle DNN'ler aracılığıyla, birçok alanda kullanılan karmaşık sinyallerin ve verilerin daha etkili bir şekilde işlenmesine olanak tanımaktadır. Bu entegrasyon, modellerin eğitim ve optimizasyonunu geliştirirken performanslarını ve doğruluklarını önemli ölçüde arttırmaktadır (Lu vd., 2022). Yapay zekânın karmaşık yapısı, doğrulama çalışmalarında büyük zorluklar ortaya çıkarmaktadır. Bu sistemlerin artan karmaşıklığı, yazılım mühendisliğinde kalite güvencesi açısından yeni zorlukları da beraberinde getirmektedir (Felderer vd., 2021). Geleneksel yazılım test yöntemleri, DNN modellerinin olasılıksal doğasına ve geniş parametre seçeneklerine uygun olmadığından, DNN'lerin değerlendirmesinde yetersiz kalmaktadır. Modellerdeki olası hatalar modelin emniyetini, güvenirliğini ve performans metriklerini etkileyebilmektedir (Moussa vd., 2024). Mevcut yazılım test araçları, hata tespiti ve test gereksinimlerini tam anlamıyla karşılamakta yetersiz kalmaktadır (Barr vd., 2015). Bu nedenle ML tabanlı sistemlerin testi, hâlâ çözülememiş bir sorun olarak varlığını sürdürmekte ve üzerinde çalışılmaya devam edilmektedir (Braiek ve Khomh, 2020). Ayrıca AI testi doğacak hataların ciddi sonuçları olabileceğinden dolayı bağımsızlık ve şeffaflık ilkelerine uygun olması gerekmektedir (Numan, 2020). Bu nedenle, DNN tabanlı sistemlerin doğrulanmasındaki zorluklar ve yöntem arayışları, günümüzde önemli bir araştırma konusudur (Sbai, 2025).

Yetersiz test yöntemleri ve uygun araçların eksikliği nedeniyle, ML tabanlı sistemler test edilmesi zor sistemlerdir ve bu nedenle "test edilemeyen programlar" kategorisine dahil edilebilmektedir (Marijan vd., 2019; Weyuker, 1982). Geçmişte, test edilemeyen ya da test edilmesi güç yazılım sistemleri için araştırmacılar tarafından pseudo-oracle yöntemleri kullanılmıştır (Davis vd., 1981). Ancak bu yöntemler, yüksek maliyetli olmalarının yanı sıra hata riskine de açıktır (Marijan vd., 2019). Özellikle büyük veri setleri üzerinde çalışan makine öğrenimi tabanlı modellerde, hatasızlığa yakın yüksek doğruluk seviyelerine ulaşmak, işlem verimliliği ile dikkatli bir denge kurmayı zorunlu kılmaktadır (Baeza-Yates vd., 2017). Bu sebeple, yeni test yöntemleri ve araçlarının geliştirilmesi kritik bir gereklilik olarak değerlendirilmektedir. Araştırmacılar bu ihtiyacı karşılamak amacıyla çeşitli test

yöntemleri geliştirmiş olsa da mutasyon tabanlı test; küçük hataları tespit edebilme ve sistemlerin değişiklikler karşısındaki hata toleransını ortaya çıkarma yeteneği ile ön plana çıkmaktadır (Jia ve Harman, 2011).

Mutasyon tabanlı yazılım testi, son otuz yılı aşkın süredir yaygın bir şekilde kullanılmaktadır (Jia ve Harman, 2011). Yöntemler geliştikçe, farklı test tekniklerine ihtiyaç duyulmakta ve mutasyon testindeki metodolojiler çeşitlenerek gelişmektedir. Yapay zekâ teknolojilerinin gelişimiyle birlikte, test süreçlerinde çeşitli zorluklar ortaya çıkmıştır. Mutasyon test yaklaşımlarıyla çözüm öneren son çalışmalar önem kazanmaktadır. Beyaz kutu ve kara kutu gibi farklı test yöntemleri geliştirilmiştir. Beyaz kutu mutasyon testi, kaynak koduna tam erişim sağlar ve koddaki değişikliklerin (mutasyonların) sistem davranışını nasıl etkilediğini test etmektedir. Öte yandan, kara kutu mutasyon testi, sistemin iç yapısını bilmeden, sadece gözlemlenebilir dış çıktılarla mutasyonların etkilerini test incelemektedir (Henard vd., 2016). Her iki test tekniği de yapay zekâ sistemlerinde uygulanabilir, ancak beyaz kutu tekniği, tüm model yapısına erişim sağlaması ve mimari mutasyonlar uygulayabilmesi nedeniyle ön plana çıkmaktadır.

Mutasyon testi, küçük hataları tespit etme yeteneği sayesinde diğer test yöntemlerine kıyasla önemli avantajlar sunmaktadır. Bu nedenle, mutasyon analizi en güçlü test yeterlilik kriterlerinden biri olarak kabul edilmektedir (Petrović vd., 2022). Yapay zekâ alanında kritik bir teknoloji olan DNN'ler, test zorluklarıyla ön plana çıkmaktadır. Bu modellerin karakteristik özelliklerini belirleyen parametre seçimleridir. Parametre seçimleri, performans üzerinde etkili olmanın yanı sıra, veri setinden kaynaklanmayan hataların büyük ölçüde ortaya çıkmasında belirleyici rol oynamaktadır. Bu bağlamda, mutasyon testi, küçük yapay hataları dahi tespit edebilme kapasitesi sayesinde, DNN'lerdeki hataların belirlenmesinde etkili bir teknik olarak değerlendirilmektedir (Ma vd., 2018). Modellerin test edilmesinde birçok etken dikkate alınmalıdır. Bu etkenlerden biri de test maliyetidir. Test maliyetleri, modelin mimarisine ve kullanılan veri setine bağlı olarak değişkenlik göstermektedir (Liu vd., 2020). Karmaşık yazılım sistemlerine entegre edilen modellerin test ve doğrulama süreçlerinde karşılaşılan zorluklar, esas olarak bu sistemlerin karmaşık mimarilerinden ve eğitimde ihtiyaç duyulan geniş veri setlerinden kaynaklanmaktadır (Ramanathan vd., 2016). Bu bağlamda, genel yazılım sistemlerinde test sürecinin zorluğu, model karmaşıklığından ve doğruluğun güvenilir şekilde değerlendirilmesi gerekliliğinden

kaynaklanmaktadır. Ayrıca, makine öğrenimi tabanlı yazılım sistemlerinde ortaya çıkan hataların, sistemin geliştirme ve bakım maliyetlerini önemli ölçüde artırabildiği gözlenmiştir (Marijan vd., 2019).

Geleneksel yazılım testlerinde etkin bir şekilde kullanılan "test oracles" (test doğrulayıcısı), derin sinir ağları gibi teknolojiler için uygun değildir. Bir “test oracle”, yazılım test süreçlerinde bir test durumunun sonucunun doğru mu yoksa yanlış mı olduğunu belirlemek için kullanılan bir yöntem veya mekanizmayı ifade etmektedir. Bu mekanizmalarla, yazılımın beklenen davranışını doğrulama ve hataları tespit etme süreci daha etkili ve güvenli hale gelmektedir. Geleneksel yazılım testlerinde bile, özellikle büyük ve karmaşık yazılım sistemlerinde test durumlarını manuel olarak oluşturmak önemli zorluklar içermektedir (Akay vd., 2021). DNN tabanlı sistemlerin karmaşık ve büyük yapıları göz önünde bulundurulduğunda, bu sistemler için test araçlarına olan ihtiyaç, geleneksel yazılımlara göre çok daha fazladır. DNN tabanlı sistemlerde "test oracle" eksikliği, bu sistemlerin olasılıksal çıktılarının beklenen değerlerle doğrudan karşılaştırılmasına izin vermemesi nedeniyle test süreçlerini daha da zorlaştırmaktadır. Bu özellik, DNN tabanlı modellerinin geleneksel yazılım test paradigması ve bu yöntemlerle uyumlu olmamasına neden olmaktadır. Bu uyum sorunu, veri odaklı süreçlerle uyumlu, özgün ve yeni metodolojilerin geliştirilmesi ihtiyacını doğurmaktadır (Marijan vd., 2019). Model testi üzerine çalışmalar olmasına rağmen, farklı ölçütlerin birbiriyle ve testlerin hata bulma kapasitesiyle ilişkisi hala farklılıklar göstermektedir (Zhang vd., 2019a).

Bu tez çalışmasında sunulan Derin Mutasyon Modulü (DMM) (Çetiner vd., 2024a), modellerin parametrelerini değiştirerek farklı model varyasyonları oluşturmakta; bunlar "mutantlar" olarak adlandırılmaktadır. Mutantların farklı varyasyonları olması sayesinde, geliştiriciler, performans verimliliği yüksek olan mutantı inceleyerek, geliştirme süreçlerinde daha etkin kararlar almasına olanak sağlamaktadır. Geliştiriciler genellikle olası tüm seçenekler arasından en uygun parametreleri seçmekte zorlanabilmektedir. Modelin hatasız bir yapıda olması ve yüksek başarıda olması özellikle gerçek dünya uygulamaları gibi kritik uygulamalarda kullanılacak modeller açısından önem taşımaktadır. Bunu keşfetmenin en etkili yollarından biri, mutasyon testi ile modeli test etmektir. Bu testleri manuel olarak gerçekleştirmek ve raporlamak çok karmaşık bir süreçtir.

Bu bağlamda, DMM, Özelleştirilmiş Mutasyon Tabanlı Yazılım Hata Enjeksiyon Aracı'na (IMFIT) (Yayan vd., 2023b) ek bir modül olarak geliştirilmiştir. IMFIT, geliştirilen yazılımların hatasız olduğundan emin olma zorunluluğundan ortaya çıkan bir yazılım test aracıdır. IMFIT, geleneksel yazılımlara mutasyonlar uygulayarak yazılımın doğrulanmasını sağlamaktadır. Mutasyon testinin potansiyelini kabul ederek, bu çalışma, Pekiştirmeli Öğrenme (RL), Uzun Kısa Süreli Bellek (LSTM), Evrişimli Sinir Ağları (CNN) ve Dönüştürücü (Transformer) mimarilere sahip modellerin test ihtiyaçlarını karşılamak amacıyla "TensorFlow" çerçevesinde geliştirilen bir DNN mutasyon test modülü sunmaktadır. Bu tez çalışmasında, DMM model tabanlı test ihtiyaçlarını karşılamak üzere özel olarak geliştirilmiştir. Geliştirme süreçlerinin sonunda özelleştirilmiş testler yapılabilmesi için etkin bir kullanıcı arayüzü sunulmaktadır. Sunulan DMM, geniş bir parametre kütüphanesi ile hedef DNN modelini tarayarak mutasyona uğratılabilir parametreleri tespit etmektedir. Tespit edilen parametreler, test kütüphanesiyle bir mutasyon planı oluşturularak mutasyona uğratılmaktadır. Bu planla hedef modelin mutantları üretilmekte, modül tarafından derlenip sınıflandırılmaktadır. Sonuçlar DMM tarafından analiz edilmekte ve mutantlar performanslarına göre ayrılmaktadır. Hedef modelden daha iyi performans verenler "hayatta kaldı", daha düşük performans verenler "öldürüldü" olarak sınıflandırılmaktadır. Bazı mutantların hedef modelden daha iyi performans göstermesi, model parametrelerinin eksik ya da hatalı yapılandırıldığını gösterebilir. Bu nedenle modellerin test edilmesi kritik öneme sahiptir. Kullanıcı, hedef modelde yalnızca seçtiği parametrelerin taramasını gerçekleştirebilir ya da tüm parametreler üzerinde tam bir tarama yaparak mutasyona uygun olanları otomatik olarak belirleyebilir. Bu parametreler mutasyona uğratılarak modelin mutantları üretilmektedir. Mutantların performans değerleri ve genel mutasyon skorları gibi bilgileri içeren test raporu, test işleminin sonunda sunulmaktadır. Geliştirilen mutasyon test aracı; kendi mutasyon kütüphanesini içermesi ve geniş test kapsamı bakımından öne çıkmaktadır.

Bu tez çalışmasında geliştirilen Derin Mutasyon Modülü (DMM), "Sürdürülebilir Kentler için İleri Teknolojiler Platformu (SÜİT)" projesi kapsamında, elektrikli araç filolarının yönetimi ve emniyetli çalışmasının sağlanması amacıyla geliştirilmiştir. Bu platformun gereksinimleri doğrultusunda, elektrikli araç rotalaması, batarya sağlığı tahmini ve sistem doğrulaması gibi kritik bileşenler üzerinde çalışan modellerin test edilmesi büyük önem arz etmektedir. Bu amaçla, RL, LSTM ve Dönüştürücü (Transformer) tabanlı modeller

kullanılmıştır. RL modeli elektrikli araçların güzergâh optimizasyonu için, LSTM modeli batarya sağlığına yönelik prognostik tahminler için tercih edilmiştir. Bu modeller, platformun simülasyon ve fiziksel ortamlarında görev alarak operasyonel verimliliğin ve güvenilirliğin sağlanmasında kritik rol üstlenmektedir. Bu bağlamda, RL modeli, LSTM modeli ve Dönüştürücü tabanlı model olmak üzere üç farklı model türüne mutasyon testleri uygulanmıştır. DMM, model türüne bağlı olarak başarı metriği olarak, "accuracy", "R^2" ve "reward_max" skorlarının sonuçlarını, oluşturulan mutantların performans skorlarıyla karşılaştırarak modelleri değerlendirmektedir. Bu süreç, modellerin hatasız ve emniyetli olduğunu belirlemek için gereklidir. LSTM modeline yönelik testlerde, 5, 10, 20 ve 30 dönem (epoch) için elde edilen mutasyon skorları sırasıyla %96,61, %91,02, %71,19 ve %68,77 olarak kaydedilmiştir. RL modeli için farklı veri setleri ve dönem sayılarıyla gerçekleştirilen testlerde %93,20, %72,13, %77,47, %79,28 ve %55,74 oranlarında mutasyon skorları ölçülmüştür. Dönüştürücü (Transformer) modelinde ise 5, 10 ve 20 dönem için yapılan testlerde, sırasıyla %75,87, %76,36 ve %74,93 doğruluk oranları elde edilmiş; bu testlerde ilgili mutasyon skorları %66,80, %73,30 ve %64,90 olarak belirlenmiştir. Bu sonuçlar, modülün hedeflenen modelleri başarıyla test edebildiğini ve "hayatta kaldı" olarak adlandırılan, hedef modellerden daha iyi performans gösteren mutantlar üretebildiğini ortaya koymaktadır. Bu, araştırmacılara modellerin performansını ve emniyetini iyileştirmek için önemli veriler sunmaktadır. Modellerin uygulamalarda kullanılmadan önce bu testlerden geçirilmesi kritik olup, olası hataların en aza indirilmesini ve model başarımının en üst düzeye çıkarılmasını sağlamaktadır. DMM, LSTM, RL, CNN ve Dönüştürücü modellerini test edebilme yeteneğine sahip olup, SÜİT kapsamında olan LSTM, RL ve Dönüştürücü modellerine uygulanmıştır.

İlerleyen bölümlerde, Bölüm 2 altında mutasyon testinin sistemleri değerlendirmedeki rolü ve bu çalışmanın temelleri hakkında önemli bilgiler sunulmaktadır. Bölüm 3'te modellerin en doğru şekilde test edilmesini sağlamak için geliştirilen DMM test aracını, test ortamlarını ve çalışmanın yöntem ve kapsamı açıklanmaktadır. Ayrıca, testlerin ön hazırlıklarına ilişkin bilgiler de bu bölümde verilmektedir. Bölüm 4'te ise LSTM, RL ve Dönüştürücü (Transformer) tabanlı yapılan testler ve sonuçlarını sunulmaktadır. Bölüm 5'te ise sonuç ve öneriler bölümü bulunmaktadır.

2. LİTERATÜR ARAŞTIRMASI

Yapay zekâ tabanlı sistemlerin yazılım geliştirme süreçlerinde hızla yaygınlaşması, bu sistemlerin kullanıma hazır olmadan önce test edilmesini zorunlu kılmıştır. Özellikle Derin Sinir Ağları (DNN'ler), geleneksel yazılım test yaklaşımlarının ötesinde yöntemlerle değerlendirilmesi gereken, karmaşık ve öğrenmeye dayalı yapılar sunar. DNN'ler, makine öğrenmesinin ve özellikle derin öğrenmenin bir alt alanı olarak, karmaşık veri yapılarını öğrenmek amacıyla geliştirilmiştir. Bu nedenle, literatürde DNN'lerin test edilmesine yönelik çeşitli yöntem ve araçlar geliştirilmiştir. Bu bölümde, DNN modellerinin testine yönelik literatürde önerilen yaklaşımlar, mutasyon testinin öğrenmeye dayalı modelleme sistemlerindeki rolü ve önemi ile mevcut mutasyon test araçları detaylı olarak incelenecektir.

2.1 Derin Sinir Ağları Modelleri İçin Test Yaklaşımları

Literatürde mutasyon testi dışında DNN'ler için farklı test yöntemleri de kullanılmıştır. Genel olarak literatürdeki yöntemler DNN modellerinin kalitesini, model başarısı, verimliliği, sağlamlığı ve emniyeti açısından değerlendirmektedir. Metamorfik testler, DNN modellerinin doğruluğunu belirlemek için kullanılmaktadır. Bu yöntem, DNN modellerinin belirli girdilerde beklenen davranışı sergileyip sergilemeyeceğini test ederek model güvenliğini belirli kapsamda sağlamayı amaçlar (Ding vd., 2017). Sinir ağı sınıflandırıcıların metamorfik ilişkilerle test edilmesi yöntemi de literatürde uygulanan diğer yöntemlerden biridir. Bu yöntem, modelin tutarlı sonuçlar üretip üretmediğini belirlemek için kullanılmıştır (Li vd., 2020). Ayrıca, DNN modellerinin kalite değerlendirmesi için kapsam metriklerine dayalı testler de kullanılmış ve bu metrikler doğrultusunda modellerin doğruluk analizleri yapılmıştır (Ayubi vd., 2023). Nöron kapsamı testi de literatürde kullanılan diğer bir yöntemdir. DNN'nin işleyişini analiz etmek ve modelin her parçasının yeterince test edilip edilmediğini tespit etmek için kullanılmıştır (Yu vd., 2023). Bir diğer test yöntemi ise modellerin girdilerine uygulanan düşman saldırı testleridir. Örneğin, bir görüntü işleme modeline yanlış sınıflandırma yapması için özel hazırlanmış girdiler verilir ve dayanıklılığı test edilmiştir (Guesmi vd., 2023). Bununla birlikte, literatürde sunulan mevcut DNN test yaklaşımları genellikle sınırlı senaryoları kapsamakta ve modellerdeki

potansiyel hataları tespit etmede yetersiz kalmaktadır. Bu nedenle mutasyon testi, kapsamlı ve sistematik bir değerlendirme sağlayarak daha derinlemesine analiz imkânı sunar.

2.2 Derin Sinir Ağları Modellerinde Mutasyon Testi

Mutasyon testi, yazılım veya makine öğrenimi (ML) tabanlı sistemlerde, sisteme bilinçli değişiklikler (mutasyonlar) uygulayıp, bu değişikliklerin sistem çıktıları üzerindeki etkilerini analiz eden bir test tekniğidir (Panichella vd., 2021). Klasik mutasyon testinin temel amacı, modelin hatalara ve değişikliklere karşı ne kadar duyarlı olduğunu anlamak ve oluşturulan mutasyonların modelin çıktı sonuçlarını nasıl etkilediğini inceleyerek modeli test etmektir (Jia vd, 2011; Zhang vd., 2016). Mutasyon testi, sistemlerin mutasyonlara karşı verdikleri karakteristik tepkileri belirlemek ve bu tepkiler üzerinden modelin zayıf yönlerini ortaya çıkarmak için kullanılmaktadır. Bu süreçte sağlanan test bilgileri, derin sinir ağlarının genel performansını ve emniyetini artırmak için kritik rol oynamaktadır (Wei vd., 2022).

Mutasyon testi, diğer DNN test yaklaşımları arasında üstün olmasının birkaç önemli nedeni bulunmaktadır. Mutasyon testi, modelin hatalarını ve başarımını sistematik olarak değerlendirmesini sağlar. Diğer test yöntemleri, örneğin metamorfik veya nöron kapsamı testleri, modelin doğruluğunu ve kapsamını belli durumlar veya sınırlı ölçütlerle değerlendirmektedir. Mutasyon testi ise, modelin farklı hata türlerine karşı nasıl performans gösterdiğini daha ayrıntılı olarak inceleme imkânı sunmaktadır. Mutasyon testi geleneksel yazılım mühendisliğinde yaygın bir şekilde çalışılmış ve uygulanmış olsa da bu yaklaşım ML sistemleri alanında yenidir ve daha karmaşıktır (Klampfl vd., 2020). Geleneksel yazılımlardan farklı olarak, ML modelleri verilerden öğrenir; bu nedenle test stratejileri, yeni paradigma ile yenilenmelidir (Marijan ve Gotlieb, 2020). Araştırmacılar ve geliştiriciler, ML tabanlı yazılımları geliştirirken bu yazılımları test etme konusunda zorluklarla karşılaşmaktadır. Mutasyon testi, bu zorlukları en aza indirmek ve modellerin başarımını ve güvenirliğini artırmak için kullanılmaktadır.

DNN tabanlı modeller genellikle büyük veri kümeleri kullanılarak eğitilir ve modellerin performansı, verilerin kalitesine ve modelin ilgili görev için uygunluğuna bağlı olmaktadır. Bu durum, DNN modellerini uygulamayı zor ve karmaşık hâle getirir. Zor ve karmaşık test süreçlerini DNN modellerine uygulamak, mutasyon testi ile mümkündür.

Çünkü mutasyon testi, modellerin başarımı ve emniyeti açısından ne kadar iyi performans gösterdiğini analiz etmek için bir çerçeve sunmaktadır (Raju, 2021). DNN tabanlı modellerin doğası gereği, performansları ve model davranışı hakkında çok fazla belirsizlik bulunmaktadır. Li vd. (2022), hem sinir ağı sınıflandırıcılarının hem de derin öğrenme kütüphanelerinin testindeki belirsizliklere yönelik mutasyon tabanlı yöntemler önermiştir. Mutasyon testi, bu belirsizliği ve karmaşıklığı azaltmak ve DNN modellerinin hatalarını ve kusurlarını tanımak için stratejiler ve teknikler sunarak kritik bir araç hâline gelmektedir (Ferreira vd., 2021). Modelleri test ederken, mutasyon testinde birçok mutant üretilir ve bu mutantların performans sonuçlarına dayalı olarak gerçek model hakkında bilgiler elde edilmektedir. Mutasyon skoru, mutasyon testinde sonuçları değerlendirmek için kritik bir metriktir. Mutasyon skoru, üretilen mutantların "öldürüldü" veya "hayatta kaldı" şeklinde sınıflandırılmasından oluşur (Naeem vd., 2019). Skor ne kadar yüksekse, model o kadar iyi yapılandırılmıştır (Arasteh vd., 2022). Skor ne kadar düşükse, model performansının düşük ve kullanıma hazır olmadığı anlamı taşır (Kusharki vd., 2022). Mutasyon skoru, oluşturulan mutantlara dayalı bilgi sağladığı için, mutasyonların modelin hangi bileşenlerinde uygulandığı da büyük önem taşımaktadır. DNN tabanlı sistemlere mutasyon testi uygulayan test araçları arasında, veri kümelerine mutasyon uygulayarak modelin performansını test etmeyi ve emniyet seviyesini değerlendirmeyi amaçlayan araçlar da bulunmaktadır (Ahuja vd., 2022). Veri seti, modelin eğitimi açısından önemli olmakla birlikte, modelin özelliklerini belirleyen asıl unsurlar modelin parametreleridir. Bu nedenle, model tabanlı bir mutasyon testi aracı geliştirerek, bu araçla test uygulamak; hataları tespit etmek ve model başarımını artırmak açısından, modellerin hatasız çalışması bakımından kritik öneme sahiptir.

2.3 Derin Sinir Ağları Modellerinde Test Araçları

DNN modellerinin karmaşık yapıları, bu sistemlerin etkin bir şekilde test edilmesi için özel olarak geliştirilmiş araçlara olan ihtiyacı arttırmaktadır. Literatürde, modellerdeki hataları tespit etmek, performansı değerlendirmek ve emniyeti sağlamak amacıyla farklı test araçları önerilmiştir.

"DeepMutation" çalışması kapsamında, DNN modellerini test etmek amacıyla mutasyon teknikleri geliştirilmiştir. Bu teknikler, DNN modellerinde olası hataların

tespitinde etkili olmuştur (Ma vd., 2018). Çalışma, mutasyon testinin hataları tespit etmedeki etkinliğini göstermiştir. Yazarlar, bu yaklaşımın yapay zekânın gelişen ve genişleyen uygulamalarıyla uyumlu hâle getirilmesi gerektiğini belirtmiştir. Başka bir çalışmada, Lu vd. (2022), RL rehberliğinde çalışan, bulanıklık ve mutasyon testi çerçevesi olan "RGChaser"ı geliştirerek, öğrenme modellerinin eğitim süreçlerini test etmeye yönelik yeni bir yaklaşım sunmuştur."RGChaser", kritik öneme sahip konuşma tanıma, makine çevirisi ve otonom sürüş gibi uygulamalarda modellerin güvenilirliğini değerlendirmek için test tekniklerinin etkili bir şekilde kullanılabileceğini vurgulayan önemli bir çalışmadır. Wu vd. (2021), DNN yapılarındaki hataların test edilmesine ve hata tespitine ek olarak, mutasyon tabanlı sistemler kullanarak bu hataların otomatik olarak onarılmasını incelemiştir. Çalışmalarında, hatalı veya uygunsuz parametre seçimlerinin öngörülemez hatalara yol açabileceğini ve bu durumun yapay zekâ yazılım kalitesi alanında önemli bir zorluk teşkil ettiğini vurgulamaktadır. Ayrıca, bu tür hataları tespit etmek ve önlemek için mutasyon testi önermiştir.

"MuNN" çalışması kapsamında, derin sinir ağlarının farklı mutasyonlar altında nasıl farklı tepkiler verdiğini derinlemesine inceleyen bir mutasyon test aracı sunulmuştur (Shen vd., 2018). Çalışmada, DNN'lerin mutasyona uğratılmasının yalnızca hataları tespit etmekle kalmayıp aynı zamanda modelin yapısını daha derinlemesine anlamayı sağladığı vurgulanmıştır. Yapılan testleri iyi bir arayüz ile sunmak önem taşımaktadır. "NeuralVis", bu alanda DNN modellerinin karmaşıklığı ve belirsizliği gibi sorunlara çözüm getirmek amacıyla geliştirilmiş ve modellerin iç yapılarını görselleştirmek ve yorumlamak için sunulmuş önemli bir çalışmadır (Zhang vd., 2019b). Modelin geliştirilmesi sırasında ön bilgi kazanmak, potansiyel gerçek dünya hatalarını önlemenin en etkili yöntemlerinden biridir. TEST-INT adlı otomatik test aracı ile, modellerin güvenli kullanımını desteklemeye yönelik bir yaklaşım sunulmuştur. Geniş ölçekli model ve veri seti testlerinin manuel yöntemlerle etkin bir şekilde gerçekleştirilemeyeceği ve ML tabanlı sistemlerin otomatik test araçlarına ihtiyaç duyduğu belirtilmiştir (Özdemir ve Demir, 2021). Wei vd. (2022), mutasyon testi aracılığıyla DNN'lerin zayıf yönlerini ortaya çıkaran ve bu zayıflıkları kapatmaya yönelik bir sistem geliştirmiştir. Geliştirilen sistem, test edilen modellerin daha yüksek doğruluk ve emniyet sağlamasını amaçlamaktadır. Ayrıca, çalışmada küçük girdilerin bile modelleri dengesizleştirebileceğini ve gerçek dünya uygulamalarında kullanılan sistemlerin, özellikle otonom araçların, bu tür girdilere karşı son derece hassas olduğunu vurgulamıştır. Küçük

girdilerin modelleri dengesizleştirerek hatalara yol açabileceği durumlar gözlemlenmiştir. Bu nedenle, model parametrelerinin en doğru şekilde seçilmesi, eğitim sürecini iyileştirdiği için modellerin hatasız ve emniyetli çalışması açısından büyük önem taşımaktadır. DNN modellerinin testine yönelik geliştirilen "DeepMutation" ve "DeepMutation++", modellerin zayıf yönlerini tespit etmek ve potansiyel hataları ortaya çıkarmak amacıyla mutasyon tabanlı test araçları olarak sunulmuştur. DeepMutation, DNN modellerine uygulanmak üzere geliştirilmiş olup, toplamda 8 farklı parametre değişikliği ile modellerin hata toleransını değerlendirmektedir (Ma vd., 2018). Bu değişiklikler ağırlık değerleri, aktivasyon fonksiyonları ve sınırlı katman yapılarına odaklanır. "DeepMutation++", bu kapsamı genişleterek 12 farklı parametre değişikliği ile hem model hem de veri üzerinde mutasyonlar uygulamaktadır (Hu vd., 2019). Ancak, her iki aracın sınırlı kapsamları, sınırlı otomasyon yetenekleri ve büyük ölçekli sistemlerde uygulanabilirlikleri açısından bazı kısıtları bulunmaktadır.

DNN tabanlı modellerin mutasyonlara karşı savunmasız olabileceği ve bu tür girdilere karşı nasıl dayanıklı hâle getirilebileceği konusu, Wei vd. (2024) tarafından yapılan mutasyon testi çalışmasında ele alınmıştır. Çalışmada, modellerin yanıltıcı girdilere karşı dirençli hâle getirilmesine yönelik bir yöntem geliştirilmiştir. Ayrıca, DNN modellerinin mutasyon testi ile sistematik olarak değerlendirilmesini sağlayan ve ölçeklenebilir test süreçleri için modelin farklı yönlerini hedefleyen bir yaklaşım sunulmuştur. Bu kapsamda, Yayan ve Aşik (2023a), "Python" programlarındaki hatalardan yola çıkarak nöral makine çevirisi kullanarak mutantlar üretmeyi amaçlayan bir yöntem önermiştir. Önerilen yaklaşım, yazılım sistemlerinde hata tabanlı mutasyon üretimini hem kolaylaştırmakta hem de otomatikleştirmektedir.

DNN modellerine ve yönelik mevcut mutasyon testi yaklaşımlarının kapsamlı analizi doğrultusunda, bu tez kapsamında özgün olarak geliştirilen Derin Mutasyon Modülünün teknik altyapısı, uygulama stratejileri ve test süreçleri bir sonraki bölümde ayrıntılı olarak ele alınacaktır. Geliştirilen sistemin test ortamı, uygulanan yöntem ve modülün teknik detayları izleyen bölümde sunulacaktır.

3. MATERYAL VE YÖNTEM

Bu bölümde, geliştirilen Derin Mutasyon Modülünün nasıl çalıştığı, hangi modeller üzerinde test edildiği ve bu testlerin hangi ortamda gerçekleştirildiği ayrıntılı şekilde açıklanmaktadır. Kullanılan donanım ve yazılım altyapısı, testlerin yürütüldüğü proje ortamı, kullanılan yöntem ve mutasyon testine tabi tutulan parametreler sistematik olarak sunulmaktadır. Bu tez çalışmasında, modelin hiper parametreleri, mimari ve fonksiyonel yapılandırmaları genel anlamda 'parametre' olarak ele alınmış ve bu parametreler üzerinden mutasyon testleri gerçekleştirilmiştir. Ayrıca, testlerin nasıl planlandığı, yürütüldüğü ve elde edilen sonuçların nasıl değerlendirildiği adım adım açıklanarak, yöntemin bilimsel dayanağı ortaya konulmuştur.

3.1 Test Ortamı

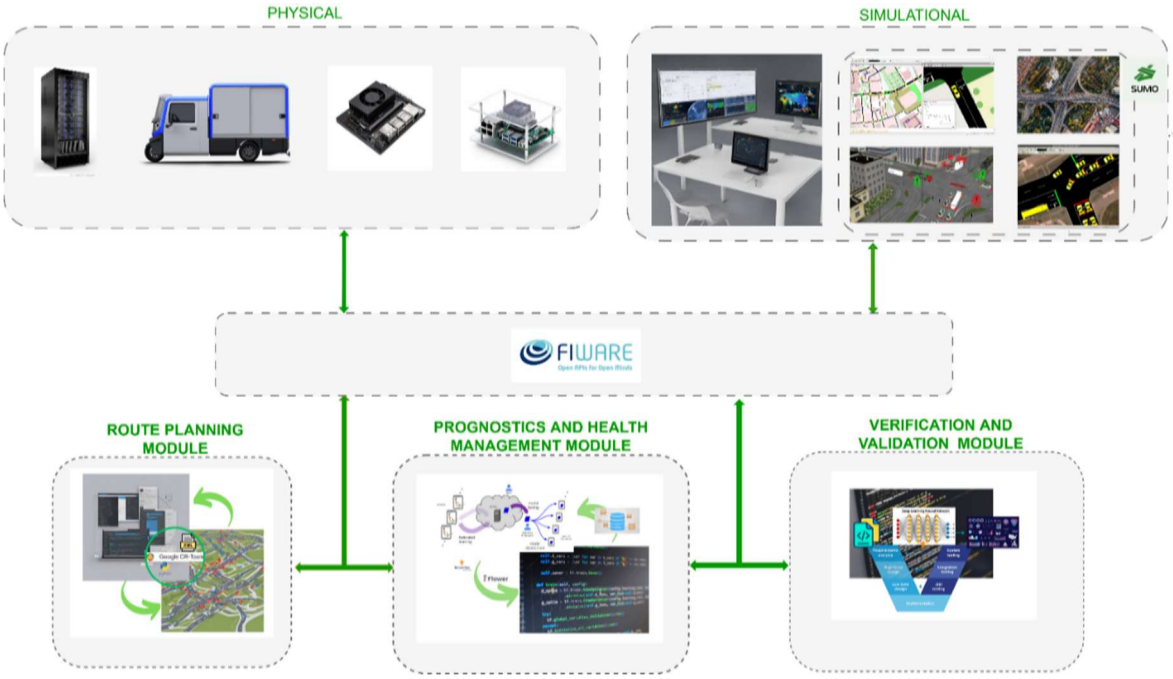
Testlerin gerçekleştirildiği bilgisayar ortamının özellikleri şu şekildedir: İşletim Sistemi: "Windows 11 Pro 64-bit (Sürüm 10.0, Derleme 22631)"; İşlemci: "12. Nesil Intel(R) Core(TM) i5-12500H (16 çekirdekli), ~2.5GHz"; RAM: "16GB"; Ekran Kartı: "NVIDIA GeForce RTX 3050 Laptop GPU". Test aracı hem Windows hem de Ubuntu sistemlerinde çalışabilmektedir.

Geliştirmeler Python programlama dilinde "MsCode" derleyicisi kullanılarak yapılmıştır. Kullanıcı arayüzü geliştirilmesi için "Qt Creator" kullanılmıştır. Testler "TensorFlow" kütüphanesini kullanan LSTM, RL ve Dönüştürücü (Transformer) tabanlı modellere yapılmıştır.

3.2 Test Proje Ortamı

Sürdürülebilir Kentler için İleri Teknolojiler Platformu kapsamındaki P6: Elektrikli Araç Filoları için Otonom Yönetim Sistemi alt projesi, Şekil 3.1'de gösterilen Otonom Filo Yönetim Kontrol Mimarisi çerçevesinde, son mil teslimatı ve batarya sağlığı tahmini amacıyla çeşitli DNN modellerinin bütünleşik olarak kullanımını kapsamaktadır (SÜİT, 2023). Bu mimari, fiziksel ve simülasyon tabanlı iki ana ortam üzerinden, rota planlama, prognostik sağlık yönetimi ve doğrulama modüllerinin "FIWARE" platformu aracılığıyla

entegre edildiği çok katmanlı bir sistemdir (FIWARE Foundation, 2023). Şekil 3.1, fiziksel ve simülasyon ortamlarının üst düzeyde, rota planlama, sağlık yönetimi ve doğrulama modüllerinin ise alt düzeyde birbirleriyle nasıl ilişkilendirildiğini göstermektedir. Fiziksel ortamda, elektrikli araçlar, sunucular ve donanım altyapıları yer almakta; simülasyon ortamında ise "Eclipse SUMO" tabanlı trafik ve rota planlama simülasyonları gerçekleştirilmektedir (DLR, 2025). Bu iki ortam, "FIWARE" aracılığıyla merkezi bir yapı altında veri akışı sağlayarak, tüm modüllerin birlikte çalışmasını mümkün kılmaktadır.



Şekil 3.1 Proje ortamı (Çetiner'den, 2024a)

Mimari kapsamında üç ana iş paketi bütünleşmiş şekilde çalışmaktadır. Birincisi, Rota Planlama İş Paketi (Route Planning Work Package), elektrikli araç filolarında rotalama problemlerine yönelik modellerin geliştirilmesinden sorumludur. Bu iş paketinde, Pekiştirmeli Öğrenme (RL) tabanlı yaklaşımlar kullanılarak rota planlama çözümleri üretilmektedir. Geliştirilen modeller, simülasyon ortamlarında uygulanarak rotalama algoritmalarının verimliliği üzerine çalışılmaktadır. İkincisi, Prognostik ve Sağlık Yönetimi İş Paketi (Prognostics and Health Management Work Package), araçların batarya sağlık durumlarının tahminine odaklanmakta ve LSTM tabanlı model geliştirerek filo operasyonlarının sürekliliği açısından destek sağlamaktadır. Prognostik ve Sağlık Yönetimi İş Paketi hem gerçek saha verileriyle hem de simülasyon ortamından elde edilen bilgilerle

beslenir ve batarya durumu tahminlerini sistemi desteklemek üzere kullanmaktadır. Üçüncü iş paketi olan Doğrulama ve Onaylama İş Paketi (Verification and Validation Work Package) ise, geliştirilen modelleri test etmek amacıyla çalışmaktadır. Bu iş paketi kapsamında, Derin Mutasyon Modülü devreye girer; mutasyon tabanlı testler uygulanarak modellerdeki potansiyel zayıf noktalar ve emniyetle ilgili sorunlar ortaya çıkarılmaktadır. Elde edilen bulgular geliştiricilere sunularak, modellerin iyileştirilmesine yönelik test sonuçları sağlanmaktadır.

Proje kapsamında geliştirilen modellerin performans değerlendirmeleri Eclipse SUMO simülasyon ortamında yapılmaktadır, bu değerlendirmeler sonucunda geliştirilen modeller Otonom Filo Yönetim Kontrol Mimarisine entegre edilmektedir. Şekil 3.1'de görülen yeşil bağlantılar, modüller arası veri akışını ve FIWARE tabanlı entegrasyonu temsil etmektedir. DMM, DNN modellerinde parametre mutasyonları gerçekleştirerek geliştiricilere test analizleri sunmaktadır. Bu süreçte, kaynak modelin başarı metriği eşik değer olarak alınmakta ve mutantlar bu eşiğe göre sınıflandırılmaktadır. Örneğin, R^2 skoru 0,9 olan modelin mutantlarının 0,98 skoru vermesi, modelin iyileştirilebilir yapıda olduğunu göstermektedir. Testlerde anormal performans dalgalanmaları gözlemlenmediği sürece ve mutasyon skoru yeterli seviyedeyse model test sürecini başarıyla tamamlamış kabul edilmektedir. Bununla birlikte, sistemin sürekli veri değişimine ve operasyonel gereksinimlere bağlı olarak yeniden değerlendirilmesi gerekebilir. DMM aracılığıyla modeller gerekli görüldüğünde tekrar test edilebilmektedir. Gerek görülürse modelde iyileştirme yapılabilmektedir.

Bu tez kapsamında geliştirilmiş olan DMM, DNN modelleri üzerinde mutasyon testleri gerçekleştirerek sürekli gelişime katkıda bulunmaktadır. Mutasyon testi sonuçları, geliştiricilerin emniyet ve performans açısından modellerini iyileştirmelerine yardımcı olmaktadır. Geliştirme ve test süreci, en iyi sonuçlar elde edilene kadar sürekli olarak sürdürülmektedir. Geliştirilen modeller Otonom Filo Yönetim Kontrol Mimarisi ile bütünleştirildiğinde, hata payı düşük olmalıdır. Çünkü büyük ve karmaşık sistemlerdeki küçük hatalar, tüm sistemi etkileyen sorunlara yol açabilmektedir. Elektrikli araç rotalamasındaki en ufak bir hata, tüm yolculuğun planlamasını olumsuz yönde etkileyebilmektedir. DMM, modellerin çeşitli mutantlarını oluşturarak model özelliklerinin daha iyi anlaşılmasını ve gerçek dünyada kullanılmadan önce olası hataların ve başarım

yeterliliğinin mutasyon testi ile açığa çıkarılmasını sağlamakta, böylece model emniyetini artırmaktadır.

Mutasyon testleri, modellerin parametrelerinde değişiklikler gerçekleştirir. Modül, ilk olarak kaynak modelin başarı metriğini eşik değer olarak kabul eder ve üretilen mutantları bu eşiğe göre sınıflandırmaktadır. Mutantların çoğu başarı metriği puanının altında kalıyorsa, modelin mutasyon skorunun düşük olduğu belirtilmektedir. Öldürüldü olarak sınıflandırılan mutantların fazlalığı nedeniyle mutasyon skoru düşük olacağından, test edilen model test sonuçlarıyla birlikte geliştiricilere iletilmeli ve model güncellenmelidir. Test sonuçlarında anormal performans düşüşleri gözlemlenirse, geliştiricilerin modeldeki hata ve hassasiyetleri tespit ederek düzeltmeleri gerekmektedir.

Mutasyon skoru istenilen seviyede ise ve mutantların performanslarında küçük değişiklikler sonucunda anormal performans sapmaları gözlemlenmiyorsa, model testi başarıyla tamamlanmış kabul edilmektedir. Sistem mükemmel çalışsa bile, modellerin değişen koşullara karşı sürekli olarak güncellenmesi gerekmektedir. Gerçek dünya uygulamalarında, değişen çevre koşullarına uyum sağlamak kritik önem taşımaktadır. Güncellenmiş ve optimize edilmiş bir sistem, eski ve güncel olmayan bir sisteme kıyasla genellikle daha emniyetli ve daha yüksek başarım düzeyine sahip olmaktadır. Bu kapsamda, DMM güncellemeler sonucunda oluşan yeni modelleri yeniden test ederek, oluşabilecek hatalar ve emniyet sorunları hakkında geliştiricilere mutasyon test analizleri sunmaktadır.

3.3 Derin Mutasyon Modülü

DMM, tarama ve yürütme gibi farklı işlemler için çeşitli test gereksinimlerini karşılamak üzere tasarlanmış bir test modülüdür. Tarama sürecinde modül, kullanıcı tarafından belirtilen kod parçacıklarına göre özelleştirilebilir taramalar gerçekleştirebilmekte veya otomatik tarama yapabilmektedir. Yürütme sürecinde, farklı metrikler tanımlanmakta ve mutant kodları analiz edilerek modelin başarı metriği puanının etkisine göre performansı hesaplanmaktadır. DMM, katmanlar ve nöronlar gibi sınırlı geleneksel mutasyon stratejilerinin ötesine geçerek daha geniş bir parametre aralığını kapsamaktadır. Bu yaklaşım, mutasyon testinin kapsamını genişletmekte ve isteğe bağlı olarak birçok parametreye mutasyonların uygulanmasını mümkün kılmaktadır. Bu da

geliştiricilere modeldeki olası hataların tespit edilmesi ve model başarımının artırılması için bilgiler sağlamaktadır. Ayrıca, model emniyetinin iyileştirilmesine katkıda bulunmaktadır.

Geleneksel yazılımlarda oluşan kritik hataları anlamak, DNN tabanlı sistemlere kıyasla daha kolay ve daha öngörülebilir bir süreçtir. DNN tabanlı sistemlerde ise, uygun koşullar oluşana kadar modeldeki bazı hataları tespit etmek mümkün olmayabilmektedir. Bu nedenle, DMM mutasyon test tekniğini kullanarak modellerin parametrelerine mutasyonlar uygulamakta ve modelin uygulamalarda kullanıma hazır olup olmadığı ile geliştirme süreçlerinin yeterliliğini belirlemeye yönelik önemli bilgiler sağlamaktadır.

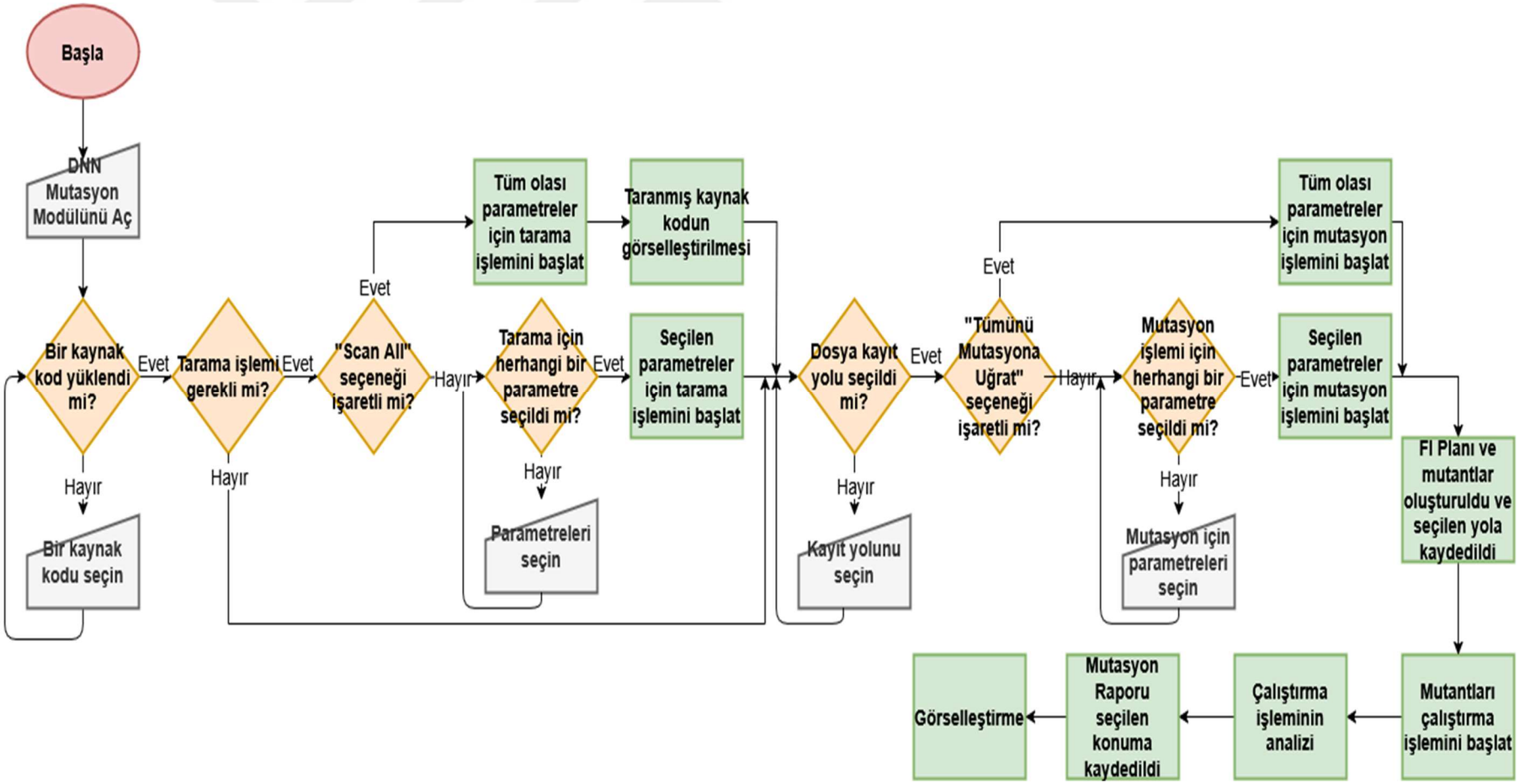
DMM, geleneksel uygulamalardaki yazılımların gerçek dünyada Makine Öğrenmesi modellerine dayalı sistemlerle değiştirilmesiyle ortaya çıkan test araçları eksikliğini gidermek amacıyla geliştirilmiştir. DNN tabanlı sistemler, çok daha karmaşık ve öngörülemez davranışlara sahip olmaktadır. Bu nedenle, klasik test yöntemleri ve araçları DNN tabanlı modellerin testinde yetersiz kalmaktadır. DMM, DNN tabanlı sistemleri test etme yeteneği ile yeni paradigmayı izleyen bir mutasyon test tekniği sunmaktadır.

3.3.1 Önerilen derin mutasyon modülü yöntemi

Geleneksel testlerde, mutasyon testi bir yazılımın test süreçlerinin, program kodundaki olası hataları belirleme yeteneğini değerlendirmek için kullanılmaktadır. Mutasyon, orijinal kod parçalarını değiştirerek çeşitli hatalar ve farklılıklar oluşturmaktadır. Bir mutant, orijinalden farklı olan ve değişiklik uygulanmış yazılım kodunu ifade eder (Naeem vd., 2019). DNN modellerinin ne kadar emniyetli olduğunu değerlendirmek için kullanılan yöntemlerden biri de mutasyon tabanlı testtir. Bu yöntem, modele bilinçli olarak değişiklikler yaparak modelin bu değişikliklere nasıl tepki verdiğini gözlemlemeye dayanmaktadır. Amaç, modelin davranışını incelemek ve olası zayıf noktalarını ile hatalarını ortaya çıkarmaktır. Yapılan her değişiklik sonrasında oluşan mutantların çıktıları analiz edilmekte ve sınıflandırılmaktadır. Bu sınıflandırma sonucunda bir mutasyon skoru hesaplanmaktadır. Mutasyon skoru, testlerin modeli ne kadar etkili bir şekilde sınayabildiğini gösteren bir ölçüttür. Model, ne kadar farklı durumda test edilirse, emniyeti hakkında o kadar fazla bilgi elde edilmesi mümkün olmaktadır.

DMM, mutasyon kütüphanesini kullanarak, modellerin yapılarında belirlenen parametrelere mutasyonlar uygulamak suretiyle mutantlar üretmektedir. Bu mutasyon kütüphanesi, geliştiriciler tarafından model yapılandırmaları üzerine yapılan literatür araştırmalarından oluşturulmuştur. Hedef modelde rastgele değişiklikler yapılmak yerine, mutasyonlar bu kütüphane aracılığıyla uygulanmaktadır. Oluşturulan mutantlar, orijinal kaynak kodunun çıktıları ile karşılaştırılmaktadır. Mutantın çıktı değerleri, orijinal kaynak kodundan elde edilen çıktının eşik değerine ulaşmazsa, "öldürüldü" olarak sınıflandırılmakta; eşik değerine ulaşır veya aşarsa, "hayatta kaldı" olarak sınıflandırılmaktadır. Modül, hedef modelde mutasyona uğratılabilecek parametreleri tarama yoluyla belirlemekte ve buna göre mutant üretimi için bir plan oluşturmaktadır. Bu plana göre, tüm mutantlar oluşturulmakta ve mutasyon planıyla birlikte kullanıcı tarafından belirtilen klasöre kaydedilmektedir. Oluşturulan mutantlar derlenmekte ve sonuçlar kayıt altına alınmaktadır. Bu sonuçlar hesaplanarak bir mutasyon puanı belirlenmekte ve modelin test süreci tamamlanmaktadır. Test edilen model, oluşturulan test raporuna göre değerlendirilmektedir. Tüm bu süreç, Şekil 3.2'de akış şeması olarak gösterilmektedir. Ortaya çıkan sonuçlar, DMM tarafından test edilen modelin mutasyon test sonuçları olarak sunulmaktadır.

Testler, farklı model türlerine uygulanabilmektedir. Modeller aynı yapıları kullanmadığından, her model türü için kullanılan performans metrikleri farklılık göstermektedir. Çizelge 3.1'de görüldüğü üzere, DMM Uzun-Kısa Süreli Bellek Ağları (LSTM), Pekiştirmeli Öğrenme (RL) ve Dönüştürücü (Transformer) modellerine ait mutantları, farklı performans metriklerini esas alarak sınıflandırmaktadır. LSTM modellerinde modül, R^2 puanını kullanarak mutasyona uğramış modelin performansını, hedef modelin performans metriği ile karşılaştırmaktadır. Hedef modelin performansı, eşik değer olarak kabul edilmektedir. Mutant modelde R^2 puanında bir düşüş gözlemlendiğinde mutant "öldürüldü", artış gözlemlendiğinde ise "hayatta kaldı" olarak sınıflandırılmaktadır.



Şekil 3.2. Derin Mutasyon Modülü Akış Diyagramı (Çetiner'den, 2024a)

Pekiştirmeli Öğrenme modellerinde ise metrikler genellikle modele bağlı olarak değişiklik göstermektedir. Elektrikli araçlar için en uygun rotayı belirlemede kullanılan “reward_max” metriği esas alınmakta olup, bu metrikteki azalma mutantın "öldürüldü", artış ise "hayatta kaldı" olarak sınıflandırılmasına neden olmaktadır. Dönüştürücü (Transformer) modellerinde ise doğruluk (accuracy) metriği kullanılmaktadır. Düşük doğruluk değerine sahip mutantlar "öldürüldü", yüksek doğruluk değerine sahip olanlar "hayatta kaldı" olarak sınıflandırılmaktadır.

Çizelge 3.1. Model tipine göre değerlendirme koşulları

Model Tipi	**Performans Metriği**	**Öldürme Koşulu**	**Hayatta Kalma Koşulu**
LSTM Model	R^2 Score	düşük	yüksek
RL Model	reward_max	yüksek	düşük
Transformer Model	accuracy	düşük	yüksek

Bu yöntemler, modülün çeşitli DNN modelleri üzerinde etkili bir şekilde test yapabilmesini ve her model türü için özelleştirilmiş bir mutasyon testi yaklaşımı sunabilmesini sağlamaktadır. Mutasyon testine başlamadan önce, DMM kullanıcı arayüzünde hedef modelin tipi seçilmekte ve testlerin modele uygun şekilde uygulanması sağlanmaktadır.

Parametrelere uygulanan mutasyonlar, modülün mutasyon kütüphanesinden alınmaktadır. Mutasyon kütüphanesinde her parametre için önceden belirlenmiş özel mutasyonlar bulunmaktadır. Bu mutasyonlar, her parametre için özgün olup farklılık göstermektedir. Modül, tarama süreci sonucunda tespit ettiği ilgili parametreyi değiştirmek amacıyla, o parametreye özgü hazırlanmış değerleri kullanmaktadır. Hedef parametre, aktivasyon fonksiyonu gibi sınırlı sayıda değer alabiliyorsa, mümkün olan tüm değerler için mutantlar oluşturulmaktadır. Kütüphane, temel bir mutasyon yaklaşımı sunmakta olup, test maliyeti ve test kapsamına bağlı olarak güncellenebilir şekilde tasarlanmıştır. Hedef modele göre, test öncesinde uzman görüşleri doğrultusunda mutasyonlar güncellenerek test kapsamı arttırılabilmekte veya test maliyetini düşürmek amacıyla testler sınırlandırılabilmektedir.

Test sonuçlarını değerlendirmek için mutasyon skoru kullanılmaktadır. Mutasyon skoru, test edilen modelin "hayatta kaldı" ve "öldürüldü" olarak sınıflandırılan mutant sayısına göre hesaplanan bir başarı puanıdır. Öldürülen mutantlar "K", mutasyon skoru "M" ve toplam mutant sayısı "N" olarak ifade edilmektedir. Mutasyon skoru, öldürülen mutant sayısının toplam mutant sayısına bölünmesi ve Denklem 3.1'de gösterildiği gibi 100 ile çarpılmasıyla elde edilmektedir.

Denklem 3.1, DNN mutasyon testi bağlamında kritik bir ölçüt olan mutasyon skorunu hesaplamak için kullanılır. Bu skor, testteki modellerin performansını gösterir ve ne kadar yüksek olursa, model o kadar hatasız ve emniyetli olur. Mutasyon skoru, test süreci sırasında başarıyla "öldürüldü" mutantların yüzdesini ifade etmektedir.

$$M = 100 * \frac{K}{N} \tag{3.1}$$

Mutasyon skorunun, mutasyon testlerinde kritik bir başarı ölçütü olarak kullanılması, test süreçlerinin etkinliğini ve kapsamlılığını derinlemesine değerlendirmeyi mümkün kılmaktadır. Mutasyon skoru, DMM'nin mutantları başarıyla sınıflandırması sonucunda hesaplanmaktadır. Bu skor, modelin hatasız ve uygulamalara hazır olup olmadığının değerlendirilmesi açısından önem arz etmektedir. Yüksek mutasyon skoruna sahip modellerin, daha düşük skora sahip modellere kıyasla daha yüksek emniyet ve başarım düzeyi sunduğu değerlendirilmektedir. Özellikle, modellerin gerçek dünya uygulamaları gibi kritik alanlarda kullanılmadan önce yüksek mutasyon skoru sağlaması beklenmektedir.

Testlerin işleyiş algoritması, Şekil 3.3'te gösterildiği üzere, hedef modelin yüklenmesi ve başarı metriğinin hesaplanması ile, modelin mutantlarının üretilmesiyle başlamaktadır. Üretilen mutantlar tek tek derlenmekte ve her birine ait performans metrikleri kaydedilmektedir. Daha sonra mutantlar, elde edilen performans sonuçlarına göre sınıflandırılmaktadır. Sınıflandırma sonucunda mutasyon skoru elde edilmektedir. Ayrıntılı test raporu, belirlenen dosya yoluna kaydedilmekte ve mutasyon testi sonuçları kullanıcı arayüzü üzerinden incelenebilmektedir.

Test Oracle
Girdi: OrijinalModeller
Çıktı: MutasyonSkoru μ
1: Orijinal modelin türünü belirle
2: MinMetric, eğitilmiş orijinal modelin başarı eşik metriği olsun
3: MutantTestMetric, bir mutantın test metriği olsun
4: m, Deep Mutation Modülü tarafından oluşturulan toplam mutant sayısını ifade etsin
5: k, öldürülen mutantların sayısı olsun
6: MinMetric = ComputeMinMetric(OrijinalModel)
7: Model üzerinde DNN modülünü çalıştırarak mutantları oluştur
8: yeni mutant olduğu sürece
9: TrainMutant()
10: MutantTestMetric = EvaluateMetric() olarak belirle
11: Eğer MutantTestMetric > MinMetric ise
12: Mutant hayatta kalır
13: aksi halde
14: Mutant öldürülür
15: k = k+1 olarak güncelle
16: end if
17: end while
18: Öldürülen mutant sayısı (k) ve hayatta kalan mutant sayısı (m-k) belirlenir
20: Mutasyon skoru (μ) şu formül ile hesaplanır: μ = k / m

Şekil 3.3. Derin Mutasyon Modülü mutasyon algoritması

Bu tez çalışmasında sunulan parametreler, Çizelge 3.2'te gösterildiği üzere dört temel kategori altında sınıflandırılmaktadır: optimizasyon, sinir ağı mimarisi, ağırlık başlatma ve düzenleme ve eğitim kontrolü. Optimizasyon kategorisinde, "learning_rate", "optimizer", "momentum","epsilon", "decay", "global_step", "decay_steps" ve "decay_rate" gibi parametreler yer almaktadır. Bu parametreler, modelin öğrenme sürecini doğrudan etkileyen bileşenler olup, eğitim sırasında ağırlıkların güncellenmesinde kullanılmaktadır. Sinir ağı mimarisi başlığı altında değerlendirilen parametreler ise "activation", "dropout", "filters", "strides", "padding", "dilation_rate", "units", "input_shape", "ff_dim", "maxlen", "vocab_size", "num_heads", "embed_dim", "keep_prob", "rate", "ksize", "groups", "axis" ve "depth_radius" gibi modelin yapısal özelliklerini belirleyen parametreleri içermektedir. Ağırlık başlatma ve düzenleme kategorisinde yer alan "bias_initializer", "beta_initializer", "gamma_initializer", "moving_mean_initializer", "moving_variance_initializer", "kernel_regularizer", "bias_regularizer", ve "bias_constraint" gibi parametreler, ağırlıkların başlangıç değerlerini tanımlamak ve aşırı öğrenmeyi önlemeye yönelik düzenleme yöntemlerini temsil etmektedir. Eğitim kontrolü kapsamında değerlendirilen parametreler ise "loss",

“from_logits”, “training”, “seed”, “label_smoothing”, “batch_size”, “capacity” ve “max_to_keep” gibi parametrelerden oluşmaktadır. Bu parametreler, modelin eğitim davranışını kontrol etmek, veri akışını yönetmek ve boyutsal-yapısal konfigürasyonları ayarlamak amacıyla kullanılmaktadır. Tüm parametreler ve modül kapsamı, Çizelge 3.2’de ayrıntılı olarak sunulmaktadır.

Geliştirilen Derin Mutasyon Modülü, “TensorFlow” kütüphanesi ve mevcut literatürdeki mutasyon testi yaklaşımlarına uygun olarak tasarlanmış geniş kapsamlı bir mutasyon kütüphanesine dayanmaktadır. LSTM, RL ve Dönüştürücü (Transformer) tabanlı modellerde gerçekleştirilen mutasyon testlerinde kullanılan parametreler, modül tarafından yürütülen tarama süreci ile otomatik olarak tespit edilmekte ve test maliyetleri ile bu parametrelerin model davranışı üzerindeki olası etkileri dikkate alınarak uygun olanlar seçilmektedir.

Test modülü, Çizelge 3.2’de gösterildiği üzere yaklaşık 50’den fazla parametreyi mutasyona uğratabilmektedir. Bu durum, araştırmacıların ve geliştiricilerin DNN'leri çeşitli bakış açılarından test etmelerine olanak sağlamaktadır. DMM ayrıca hedef modelin “Python” kodunu yükleme, mutasyon işlemlerini tarama ve doğrudan mutasyon uygulama işlemlerini desteklemektedir. Bu özellikler, “TensorFlow” çerçevesi için özel olarak tasarlanmıştır.

DMM kapsamı, “TensorFlow” DNN çerçevesinde kullanılan önemli parametreleri içerecek şekilde tasarlanmıştır. Bu parametrelere uygulanan mutasyonlar, modülün mutasyon kütüphanesi olan kaynaktan seçilmektedir. Üretilen mutantların sayısı, parametreye bağlı olarak değişiklik göstermektedir. Modül kapsamı güncellenebilir ve arttırabilir olarak geliştirilmiştir. Değişen ve gelişen yöntemler, test maliyetleri ve model türleri gibi birçok farklı değişken açısından testlerin özelleştirilebilmesi önemlidir. Parametre kapsamı literatürdeki empirik yaklaşımlara temel olabilecek şekilde sunulmuştur.

Kütüphanedeki bazı parametreler istenirse fazla sayıda değer alabilmektedir. Örneğin, “epsilon” gibi parametreler fazla sayıda mutasyon için uygun olabilmektedir. Testin amacı, yeterli kapsamda mutasyonlar üretmek ve test maliyetini göz önünde bulundurmaktır. Bu nedenle, mutasyon kütüphanesi, en olağan ve hata eğilimli gibi önceden

belirlenmiş değerler içermektedir. Bu değerler, test maliyeti ile test kapsamı arasındaki dengeyi sağlamak üzere seçilmiştir. Testlerden önce bu değerler güncellenerek, test kapsamı artırılabilir veya test maliyeti azaltılabilir.

Çizelge 3.2. Modül kütüphanesi parametre kapsamları

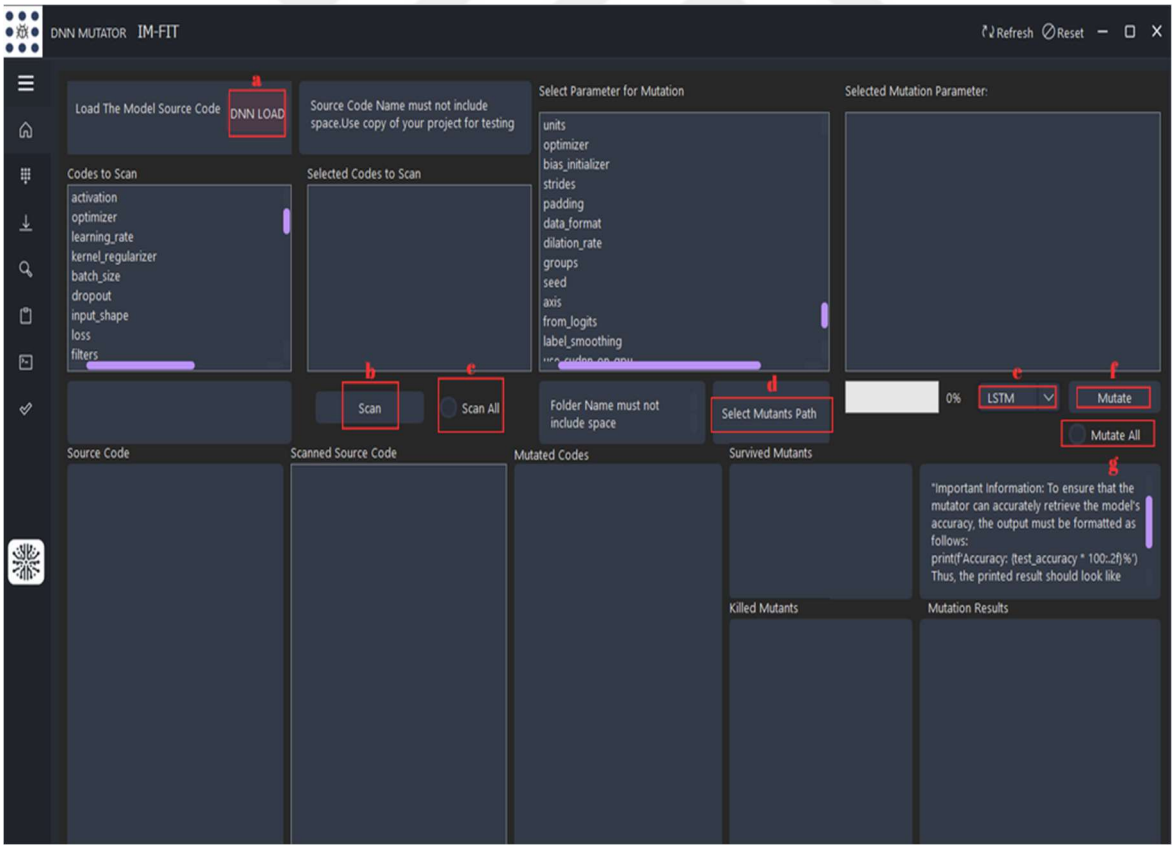
Parametre Kategorileri	**Parametreler**	**Uygulanabilir Mutasyonlar**	**Mutasyon Test Türü**
Optimizasyon	optimizer	-Olası değerler -Hatalar/Boş değer ataması	Durum Mutasyonları
Optimizasyon	momentum, beta1, beta2, epsilon,decay, decay_steps, decay_rate,global_step, epsilon_decay, min_epsilon learning_rate	-Olası değerler -Hatalar/Boş değer ataması	Değer Mutasyonları,
Ağ Mimarisi	activation	-Olası değerler -Hatalar/Boş değer ataması	Durum Mutasyonları
Ağ Mimarisi	dropout, filters, strides, padding, dilation_rate,axis, depth_radius groups,units,input_shape,embed_dim, ff_dim,maxlen,vocab_size,num_heads, keep_prob,rate,ksize	-Olası değerler - Hatalar/Boş değer ataması	Değer Mutasyonları
Başlatma ve Düzenleme	kernel_initializer, bias_initializer, bias, beta_initializer, bias_regularizer, bias_constraint, gamma_initializer, moving_mean_initializer, moving_variance_initializer, epsilon, kernel_regularizer ,l1, l2	-Olası değerler - Hatalar/Boş değer ataması	Değer Mutasyonları
Eğitim Kontrolü	loss,from_logits, training, center, scale, trainable,	-Olası değerler - Hatalar/Boş değer ataması	Durum Mutasyonları
Eğitim Kontrolü	capacity, seed, label_smoothing, batch_size, decay, max_to_keep	-Olası değerler - Hatalar/Boş değer ataması	Değer Mutasyonları

3.3.2 Derin mutasyon modülü tarama işlemi

Derin Mutasyon Modülünün geliştirilmesi sürecinde, tarama süreci modülün kritik ve işlevsel aşamalarından birini oluşturmaktadır. Bu süreç, modülün DNN tabanlı modellerin yapısal ve işlevsel bileşenlerini ayrıntılı olarak incelemesine olanak tanımakta ve

kullanıcıya potansiyel olarak hataya açık kod satırlarını belirlemektedir. Tarama aşamasında, DMM tarafından tespit edilen değiştirilebilir kod parçaları, test sürecine alınmadan önce kullanıcıya sunulmaktadır. Bu amaçla modellerin kritik bileşenleri belirli parametreler açısından taranmaktadır.

Kullanıcı, belirli kod parçalarını seçerek Şekil 3.4'teki tarama işlemini özelleştirebilir veya modül tüm değiştirilebilir değerleri otomatik olarak taranmasını sağlayabilmektedir. Kullanıcılar, tüm tarama özelliğini kullanarak hızlı ve güvenilir bir tarama işlemi gerçekleştirebilmektedir. Tarama işlemi, kullanıcıların kaynak kodunun dosya yolunu belirtmesine ve bunu modüle yüklemesine olanak tanıyan "DNN LOAD" Şekil 3.4(a) düğmesini kullanarak başlatılmaktadır. Bu aşamadan sonra, tarama işlemi iki farklı şekilde yürütülebilmektedir: Kullanıcılar genel bir tarama için "Scan All" Şekil 3.4(c)'yi seçebilir veya "Codes to Scan" bölümünden belirli parametreleri seçerek özelleştirilmiş bir tarama gerçekleştirebilirler. Tarama işlemi, "Scan" Şekil 3.4(b) etkinleştirilerek tamamlanmaktadır.



Şekil 3.4. Derin Mutasyon Modülü Kullanıcı Arayüzü

3.3.3 Derin mutasyon modülü test yürütme işlemi

Derin Mutasyon Modülü, DNN tabanlı modellerinin mutasyon testinin yapılması için kritik bir süreç olan yürütme aşamasını kapsamlı bir şekilde ele almaktadır. Mutasyon test işlemi, DNN'lerde belirli mutasyon testlerinin çalıştırılmasını ve bu testlerin sonuçlarının analiz edilmesini içermektedir.

Yürütme işlemi başlatılmadan önce hedef modelin kaynak kodu yüklenmelidir. Ayrıca, mutasyon enjeksiyon planını, mutantları ve test sonuçları verilerini kaydetmek için hedef dosya yolu, Şekil 3.4(d)'deki “Select Mutants Path” düğmesi aracılığıyla belirlenmektedir. Hedef metrikler, Şekil 3.4(e)'den hedef model türü seçilerek tanımlanmaktadır. Özelleştirilmiş mutasyon testi için parametreler, “Select Parameters for Mutation” bölümünden seçilmektedir. Ardından, “Mutate” düğmesine basılarak mutasyon testi başlatılmaktadır. Şekil 3.4(f). DMM, önce kaynak kodunu tarayarak ve mutasyon planını belirtilen dosya yoluna kaydederek bir mutasyon planı oluşturmaktadır.

Hedef dosya yolunu belirlendikten sonra, model kapsamlı test edilmek isteniyorsa, Şekil 3.4(g)'deki "Mutate All" seçeneğini seçilmektedir. Özelleştirilmiş mutasyon testi için, "Select Parameters for Mutation" bölümünden parametreler seçilmelidir. Daha sonra, Şekil 3.4(f) görülen "Mutate" düğmesine basılarak mutasyon testi başlatılmalıdır. DNN modülü önce kaynak kodunu tarayarak ve planı belirtilen dosya yoluna kaydederek bir mutasyon enjeksiyon planı oluşturmaktadır.

Şekil 3.5’te görüldüğü gibi mutasyon enjeksiyon planı oluşturulduktan sonra, ilk olarak orijinal hedef kaynak kodu test sonuçları elde edilmektedir. Bu işlemler tamamlandıktan sonra, mutasyon enjeksiyon planı içinde mutantlar oluşturarak mutasyon testi başlatılmaktadır. DMM her mutantı ayrı ayrı test etmektedir. Test süreçleri, hedef kaynak modelin ve mutantların başarı metriklerinin karşılaştırmalı analizini içermektedir. Bir mutantın başarı metriği, test edilen hedef modelin başarı metriğinden iyi sonuç verirse, "hayatta kaldı" olarak sınıflandırılmaktadır. Aksi takdirde, yani mutantın başarı metriği, hedef modelin başarı metriğinden iyi sonuç veremezse, "öldürüldü" olarak sınıflandırılmaktadır. Test süreci tamamlandıktan sonra, tüm sonuçlar Şekil 3.3’te görülen

kullanıcı arayüzündeki "Survived Mutants", "Killed Mutants" ve "Mutation Results" bölümleri aracılığıyla sunulmaktadır. Ayrıca test sonuçları sonuçlar ve mutant ayrıntıları önceden tanımlanmış bir dosya yoluna kaydedilmektedir.


```
},
{
    "Fault": {
        "Mutant_Number": 75,
        "File_Directory": "C:/Users/Gokhan/Desktop/Transformer-Test/TransformerModel.py",
        "Source_Code": "Dropout(0.1)",
        "Mutate_Code": "Dropout(0.2)",
        "Match_Number": 1
    }
},
{
    "Fault": {
        "Mutant_Number": 76,
        "File_Directory": "C:/Users/Gokhan/Desktop/Transformer-Test/TransformerModel.py",
        "Source_Code": "Dropout(0.1)",
        "Mutate_Code": "Dropout(0.3)",
        "Match_Number": 1
    }
},
{
```


Şekil 3.5 Mutasyon Planı

3.4 Test Ön Hazırlıklar

Mutasyon test süreci sırasında, Uzun Kısa Süreli Bellek (LSTM) modelleri ve Pekiştirmeli Öğrenme (RL) için mutasyona uğrayabilecek parametreler, tarama işlemi ile tespit edilmiştir. Tarama sonucunda test için belirli parametreler seçilmiştir. Tüm parametrelerin mutasyon sayıları model tipine göre değişiklik göstermektedir. Çizelge 3.3'te gösterildiği gibi RL modeli için "gamma", "alpha", "epsilon" ve "decay" parametreleri seçilmiştir. Çizelge 3.4'te gösterildiği gibi, LSTM modeli için "units" , "loss", "optimizer", "batch_size" ve" input_shape" parametreleri seçilmiştir. Çizelge 3.5'de gösterildiği gibi Dönüştürücü (Transformer) modeli için "batch_size", "loss", "epsilon", "units", "rate", "optimizer", "maxlen", "embed_dim", "num_heads" ve "ff_dim" parametreleri seçilmiştir.

LSTM tabanlı modellerin test sürecinde, modelin yapısal ve işlevsel bütünlüğünü etkileyebilecek derecede kritik öneme sahip "units", "loss", "optimizer", "batch_size" ve "input_shape" gibi parametreler üzerinde mutasyonlar uygulanmıştır. Bu parametrelerin modelin genel davranışını doğrudan şekillendirmedeki rolleri nedeniyle, yapılan

mutasyonlar aracılığıyla modelin bu tür değişimlere verdiği tepkiler analiz edilmiştir. Elde edilen mutasyon skorları ve mutantların performans çıktıları, modelin parametre düzeyindeki başarımını ve emniyetini değerlendirmek için genel bir çerçeve sunulmuştur.

Pekiştirmeli Öğrenme (RL) modellerinde ise "alpha", "epsilon", "decay(exploration_decreasing_decay)" ve "gamma" gibi temel parametreler mutasyon testine tabi tutulmuştur. Bu parametrelerin modelin öğrenme sürecindeki kritik rolü göz önüne alındığında, üzerlerinde yapılan mutasyonlar sayesinde modeli değerlendirmek için genel bir çerçeve sunulmuştur.

Dönüştürücü (Transformer) tabanlı modellerin test aşamasında ise hem mimari hem de eğitim stratejilerini etkileyen "loss", "optimizer", "batch_size", "dropout", "rate", "epsilon", "vocab_size", "num_heads", "maxlen", "embed_dim" ve "ff_dim" gibi birçok parametre kapsamlı bir şekilde değerlendirilmiştir. Bu parametreler üzerinde gerçekleştirilen mutasyonlar sonucunda elde edilen sonuçlar modelin iyileştirilebileceğini göstermektedir.

Testlerde kullanılan dönem (epoch) değerleri, test maliyeti ile test kapsamı arasında denge sağlamak amacıyla geliştiriciler tarafından belirlenen değerlere dayanarak seçilmiştir. Modelin temel dönem ve epizot değerleri, LSTM modeli için 20, RL modeli için 2500 ve Dönüştürücü (Transformer) modeli için 10 geliştiriciler tarafından olarak ayarlanmıştır. Farklı dönem değerleriyle test yapmanın avantajı, 20 dönem performansını sadece 10 dönem içinde elde edebilen mutantların olabilmektedir. Bu sebeple farklı dönem değerleriyle kapsamlı bir test amaçlanmıştır.

Çizelge 3.3. RL modeli mutasyona uğramış parametreler

Parametre	**Oluşturulan Mutant Sayısı**
alpha	29
epsilon	40
decay	30
gamma	26

Çizelge 3.4. LSTM modeli mutasyona uğramış parametreler

Parametre	**Oluşturulan Mutant Sayısı**
units(dense)	12
loss	17
optimizer	17
batch_size	13
input_shape	48
units	306

Çizelge 3.5. Dönüştürücü modeli mutasyona uğramış parametreler

Parametre	**Oluşturulan Mutant Sayısı**
loss	17
units	26
optimizer	17
batch_size	14
dropout	22
rate	8
epsilon	68
vocab_size	10
maxlen	5
embed_dim	6
num_head	5
ff_dim	5

4.BULGULAR VE TARTIŞMA

DMM, DNN tabanlı modeller için genel kullanıma uygun mutasyon testlerinin uygulanabilmesi amacıyla geliştirilmiştir. Modülün testleri; LSTM, RL ve Dönüştürücü (Transformer) modelleri üzerinde yürütülmüştür. Bu testler, RL modelinde dört farklı tipte dört parametre, LSTM modelinde altı farklı tipte yedi parametre ve Dönüştürücü modelde on iki farklı tipte on üç parametre üzerinden gerçekleştirilmiştir.

4.1 Elektrikli Araç Filoları İçin Otonom Yönetim Sistemi LSTM Modeli

SÜİT projesi kapsamında geliştirilen LSTM modeli (Yılmaz vd., 2025), elektrikli araç filolarını yönetmek için hayati şarj durumu (SoC) tahminleri yapmak üzere tasarlanmıştır. Model, gerçek dünya verilerini temsil eden NASA'nın “b0005” ve “b0006” pil veri setine kullanılarak eğitilmekte ve test edilmektedir. Modelin yapısı, modelin zaman serisi verilerindeki karmaşık bağımlılıkları öğrenmesine olanak tanıyan bir LSTM katmanı içermektedir.

LSTM modeli, elektrikli araçların şarj durumlarını tahmin ederken çeşitli gerçek dünya koşullarını ve senaryoları için kullanılabilir. Bu karmaşıklık, modelin başarısını ve hata durumlarını doğrulamayı zorunlu hale getirir. Geliştirilen mutasyon test aracı, LSTM modelinin parametrelerine ve yapısına sistematik mutasyonlar uygulayarak modelin hatalara verdiği tepkileri değerlendirmeyi amaçlamaktadır. Bu süreç, modelin zayıf noktalarını belirlemeye ve potansiyel iyileştirmeler için rehberlik sağlamaya olanak tanımaktadır.

Bu modelde, optimizasyon fonksiyonu mutasyonları (örn. 'adadelta', 'adamax', 'nadam' vb.) ve farklı nöron sayıları mutasyonları gibi farklı altı parametreye mutasyonlar uygulanmıştır. Modelin verdiği tepkiler test edilmiştir ve bu analizler, modelin emniyetine ilişkin değerlendirmelere katkı sağlamaktadır.

Kaynak kodlarının ilk taramasının ardından, seçilen parametreler DMM tarafından mutasyona uğratılarak mutantlar oluşturulur. Her mutasyon, mutant model ile hedef modelin

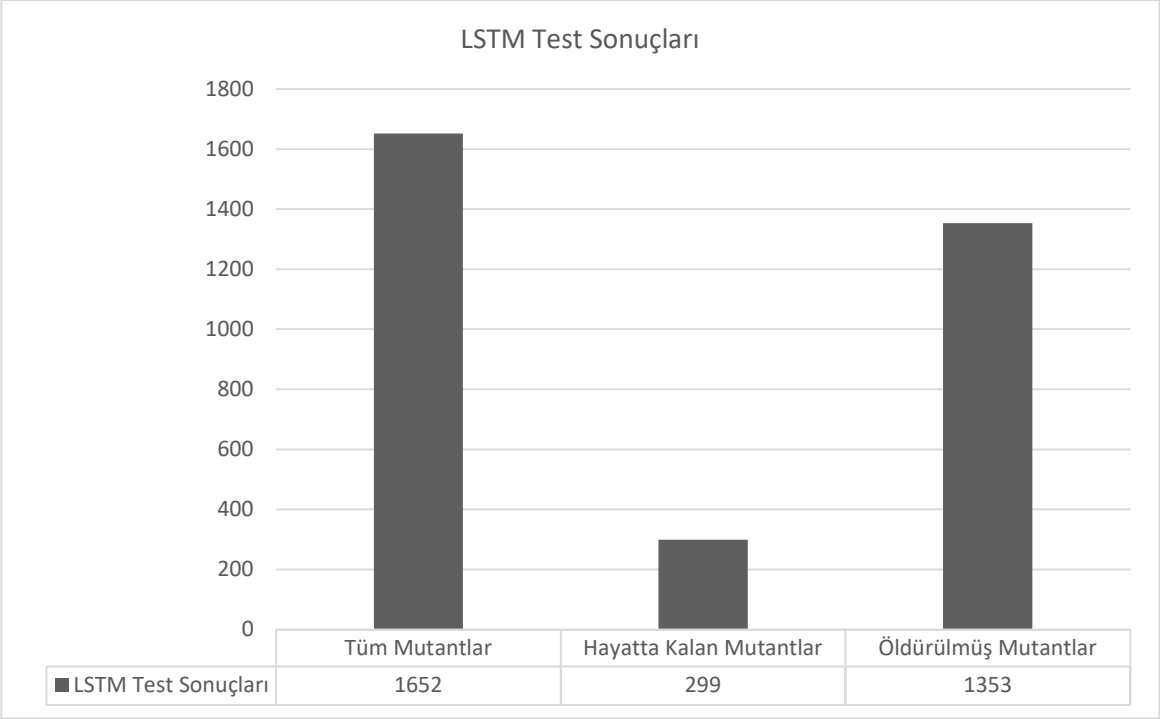
başarı metriği çıktısı arasındaki karşılaştırma yoluyla sınıflandırılır. DMM'in işlevselliği ve kod tabanında uygulanan mutasyonlar Çizelge 4.1'de örneklerle açıklanmıştır. Çizelge 4.1(a), modelin kaynak kodunu göstermektedir. Çizelge 4.1(b), LSTM modelinde "units" parametresinin mutasyon sonrası "1"den "32"ye nasıl değiştiğine dair bir örneği sunmaktadır. Ek olarak, Çizelge 4.1(c), optimizasyon fonksiyonunun "adam"dan "adagrad"a değiştiği ve "optimizer" parametresindeki değişiklikleri gösteren başka bir mutasyon senaryosunu göstermektedir. DMM tarafından gerçekleştirilen LSTM testlerinin sonuçları bu bölümde ayrıntılı olarak açıklanmıştır.

Çizelge 4.1 LSTM modeli mutasyon örneği

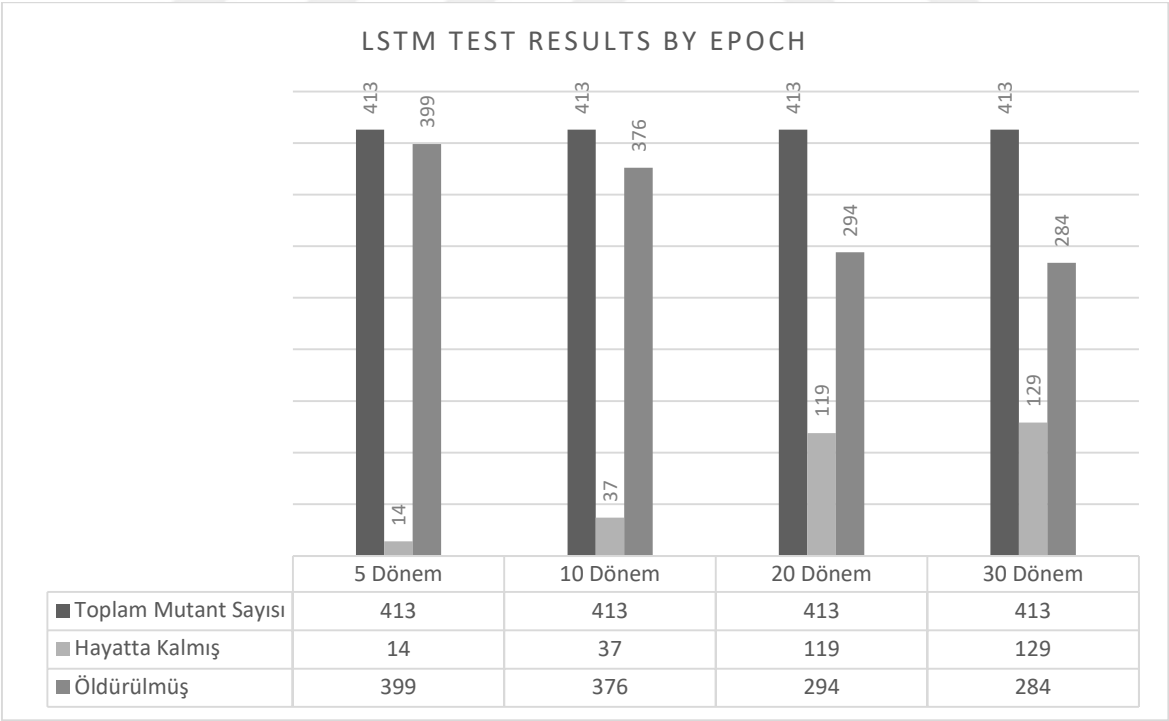
a) LSTM Model Orjinal Kaynak Kodu
model.add(Dense(units = 1)) model.compile(loss='mse', optimizer='adam')
b) Mutasyona Uğratılmış Kod (units mutated)
model.add(Dense(units=32))
c) Mutasyona Uğratılmış Kod (optimizer mutated)
model.compile(loss='mse', optimizer='adagrad')

LSTM Testinde toplam 1652 mutant ele alınmıştır. Bunlardan 299 mutant hayatta kaldı olarak sınıflandırılmıştır. 1353 mutant başarıyla "öldürüldü" olarak sınıflandırılmıştır. Sonuç olarak, LSTM modelinin DMM tarafından oluşturulan mutantlardan büyük bir oranına göre başarılı olduğunu ancak hayatta kalan mutant sayısına bakıldığında modelde iyileştirme yapılması gerektiği sonucuna varılmıştır. (Şekil 4.1).

Modelin öğrenme sürecindeki performansını anlamak için farklı dönemlerdeki (epochs) test sonuçları analiz edilmiştir. 5. dönemde 413 mutanttan 399'u öldürülmüştür ve 14'ü hayatta kalmıştır. 10. dönemde 413 mutanttan 376'sı öldürülmüştür ve 37'si hayatta kalmıştır. 20. dönemde 413 mutanttan 294'ü öldürülmüştür ve 119'u hayatta kalmıştır. 30. dönemde 413 mutanttan 284'ü öldürülmüştür ve 129'u hayatta kalmıştır. Bu sonuçlar, modelin eğitim dönemi ilerledikçe hayatta kalan mutant sayısının arttığı Şekil 4.2'de ayrıntılı bir şekilde gözlenmektedir.

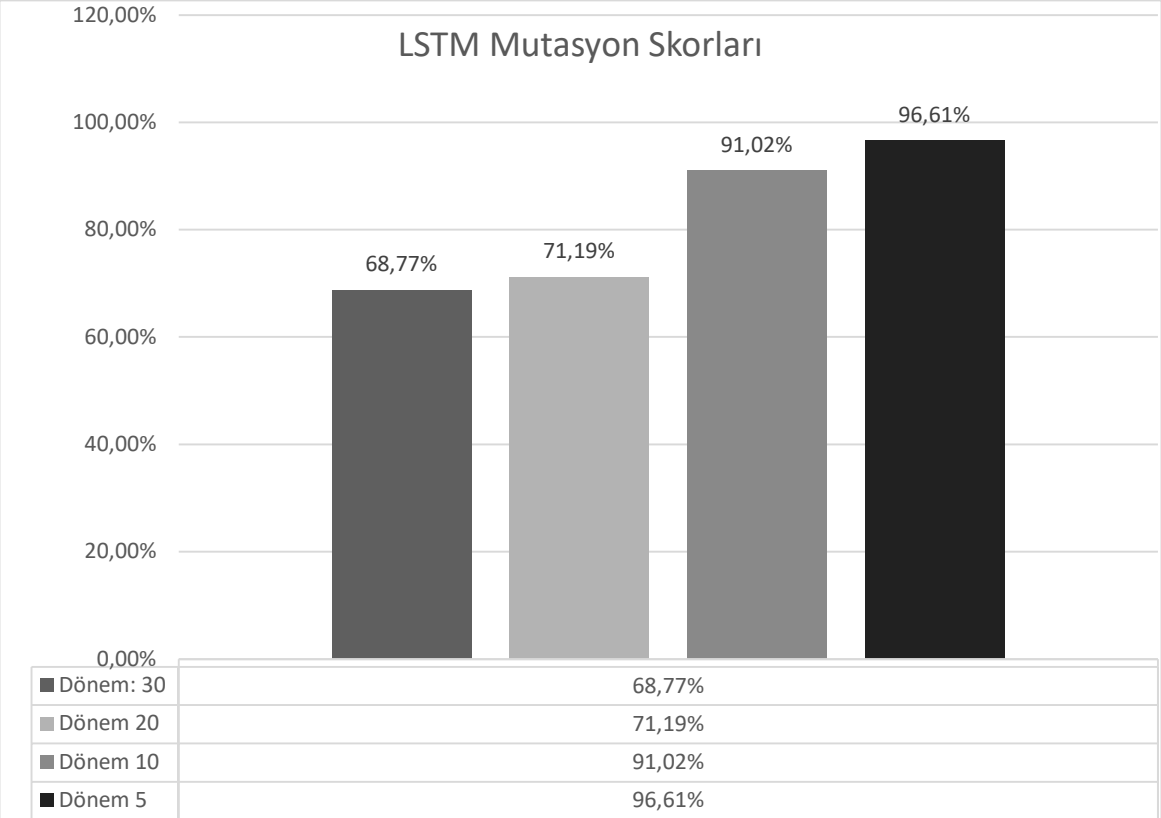


Şekil 4.1 LSTM mutasyon testi sonuçları



Şekil 4.2. LSTM dönem sayısına göre mutasyon testi sonuçları

Gerçekleştirilen testlerde, 30. dönemde (epoch) mutasyon puanı %68,77 olarak sonuçlanmıştır. 20. dönemde bu %71,19'a yükselmiştir. 10. dönemde mutasyon puanı %91,02'ye ulaşmıştır ve 5. dönemde %96,61'lik bir zirveye ulaşmıştır. Bu sonuçlar, eğitim sürecinin başında hedef modelin mutantlara karşı üstün geldiğini, ancak artan dönemle birlikte mutantların hayatta kalma oranlarının arttığı Şekil 4.3'tede ayrıntılı bir şekilde görülmektedir.



Şekil 4.3. LSTM mutasyon testi skorları

Bu sonuçlar, LSTM modelinin çeşitli mutasyonlar altında emniyet açısından yeterliliğinin değerlendirilmesini mümkün kılmaktadır. Hayatta kalan mutantların incelenmesi, modelin başarımının artırılması ve olası hataların düzeltilmesi açısından kritik öneme sahiptir. Bu testler, geliştiricilere modelin geliştirme sürecinde performans ve emniyetin en üst düzeye çıkarılabilmesi için değerli geri bildirimler sunmaktadır.

Uygulanan mutasyon testleri, modelin emniyet düzeyine ilişkin önemli veriler sağlamış ve model başarımının, gerçek dünya senaryolarında kullanılmadan önce maksimize edilmesi gerektiğini ortaya koymaktadır. Özellikle elektrikli araç filolarının şarj ihtiyaçlarını

tahmin etmek gibi emniyetin hayati önem taşıdığı kritik uygulamalarda, model testi temel bir gerekliliktir.

Modeller doğrulama süreçlerinden geçirilmeden geliştirildiğinde, geliştiriciler modelin emniyeti ve başarımı hakkında belirsizlik yaşayabilirler. Bu durum, modelin saha uygulamalarında beklenmedik hatalara karşı daha duyarlı hale gelmesine yol açabilmektedir. Yapılan testler sayesinde bu tür riskler minimize edilmekte ve modelin emniyet seviyesinin arttırılmasına katkı sağlanmaktadır.

4.2 Elektrikli Araç Filoları için Otonom Yönetim Sistemi İçin Pekiştirmeli Öğrenme (RL) ile Güzergâh Optimizasyonu

SUİT projesi kapsamında geliştirilen RL modeli, son mil teslimatında Elektrikli Araç Rotalama Problemi (EVRP) için özel olarak tasarlanmıştır. Model eğitimi için "ESOGÜ-CEVRPTW" (Aslan vd., 2025) veri seti SUİT kapsamında geliştirilmiş ve kullanılmaktadır. RL modelinin temel işlevi, müşterilerin bir yerden başka bir yere taşınmasını gerektiren alım ve bırakma işlemlerinin en verimli şekilde yürütülmesi için optimum rota planlamasını sağlamaktır. Modelin yüksek başarım ve hatasız bir şekilde ile çalışması kritik önem taşımaktadır. Bu veri seti, Eskişehir Osmangazi Üniversitesi kampüsünün gerçek coğrafi verilerini içermekte olup, şarj istasyonlarının konumları ve şarj süreleri gibi verileri içermektedir. Hedef model farklı model versiyonları üç veri kümesinde test edilmiştir: Kümelenmiş, Rastgele Kümelenmiş ve Rastgele pekiştirmeli öğrenme. Bu versiyonların her biri için 5,10,20 ve 40 müşterili olmak üzere konfigürasyonları vardır. Testlerde 1, 100, 1000, 2500, 5000 ve 10000 epizot (episode) üzerinden uygulanmıştır.

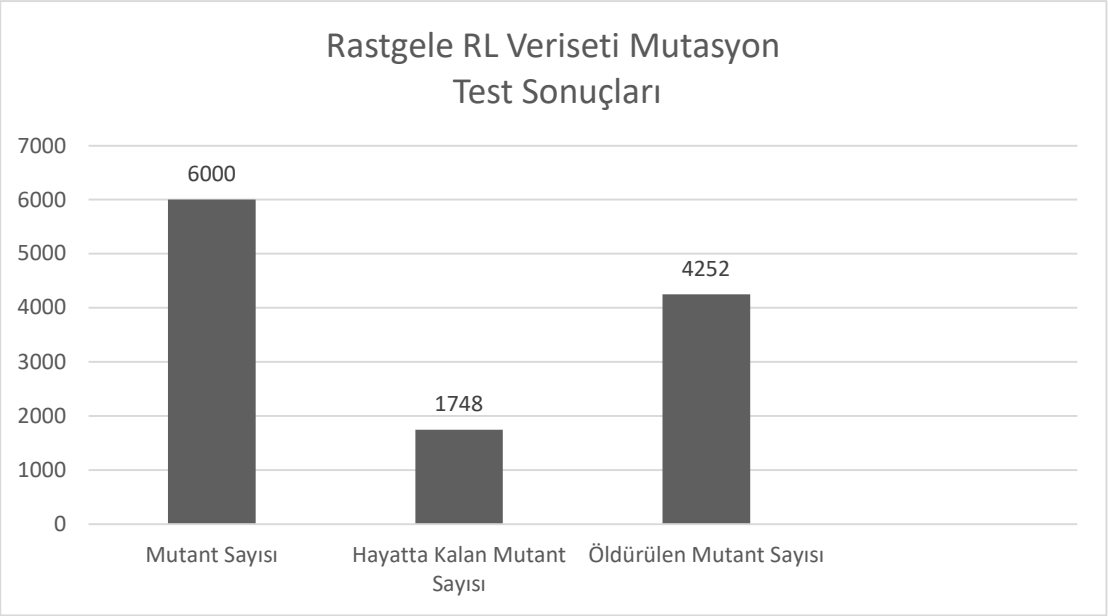
DMM'in kodlara uyguladığı mutasyon örneği Çizelge 4.2'te açıkça gösterilmiştir. RL modellerinin orijinal kaynak kodlarını içeren bölüm Çizelge 4.2(a)'da sunulmuştur. Ayrıca, Çizelge 4.2(b), modelin "exploration_decreasing_decay" parametresinin mutasyon sonucu orijinal kaynak kodundaki "0.001"den "0.0005"e değiştiği bir mutasyon örneğini göstermektedir. Benzer şekilde, Çizelge 4.2(c), mutasyon sürecinden sonra gamma parametresinin "0.99"dan "0.50"e nasıl değiştiğini gösteren başka bir mutasyon örneği sergilemektedir. Çizelge 4.2'te sunulan bu örnekler, derin mutasyon modülünün nasıl çalıştığını ve kodlara uyguladığı belirli mutasyonları göstermektedir.

Çizelge 4.2. RL modeli mutasyon örneği

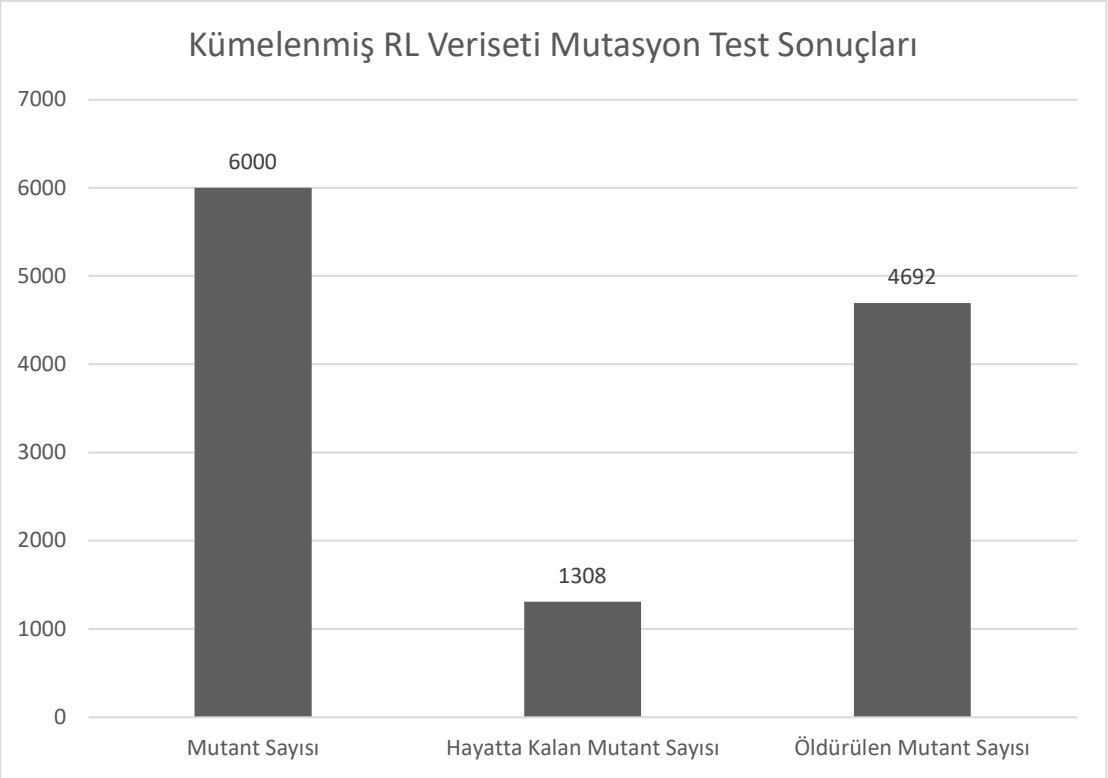
a) RL Model Hedef Parametre Örneği
gamma=0.99 exploration_decreasing_decay = 0.001
b) Mutasyona Uğratılmış Parametre Örneği
exploration_decreasing_decay=0.0005
c) Mutasyona Uğratılmış Parametre Örneği
gamma= 0.50

Mutasyon test sonuçlarını değerlendirmek, eğitim süreci boyunca modelin performansının daha derinlemesine anlaşılmasını sağlamaktadır. Şekil 4.4'te görüldüğü gibi Rastgele RL veri seti test sonuçlarında 6000 mutant üretildi bunların 4252'i öldürüldü olarak sınıflandırılmıştır. Mutantların 1748'i ise hayatta kaldı olarak sınıflandırıldı. Öldürülen ve hayatta kalan mutasyon oranlarının artan ya da azalan eğitim dönemleri (epoch) ile olan ilişkisi Çizelge 4.3'te sunulmaktadır.

Kümelenmiş RL Veri Kümeleri Mutasyon Testi Sonuçları (Şekil 4.5) için, öldürülen mutasyon sayısı 4692 ve hayatta kalan mutasyon sayısı 1308 olarak sonuçlanmıştır. Rastgele Kümelenmiş RL mutasyon testi sonuçları (Şekil 4.6) ile karşılaştırıldığında, öldürülen mutasyon sayısı 4694 ve hayatta kalan mutasyon sayısı 1306'ya düşmüştür. Rastgele ve kümelenmiş veri kümeleri için mutasyon testi sonuçlarında, her iki senaryoda da öldürülen olarak sınıflandırılan mutant sayısı yüksek olsa da yeterli olmadığı gözlenmiştir. Modelin değişikliklere modelin doğru yapılandırıldığını göstermektedir. Ancak hayatta kalan mutantlar incelenip geliştirilmesi gerektiği sonucuna varılmaktadır. Mutasyon testleri sonuçları modelin geliştiricilerine sunulmuştur. Sonuçlarla birlikte riskler minimize edilerek model emniyeti ve başarımı arttırması amaçlanmıştır.

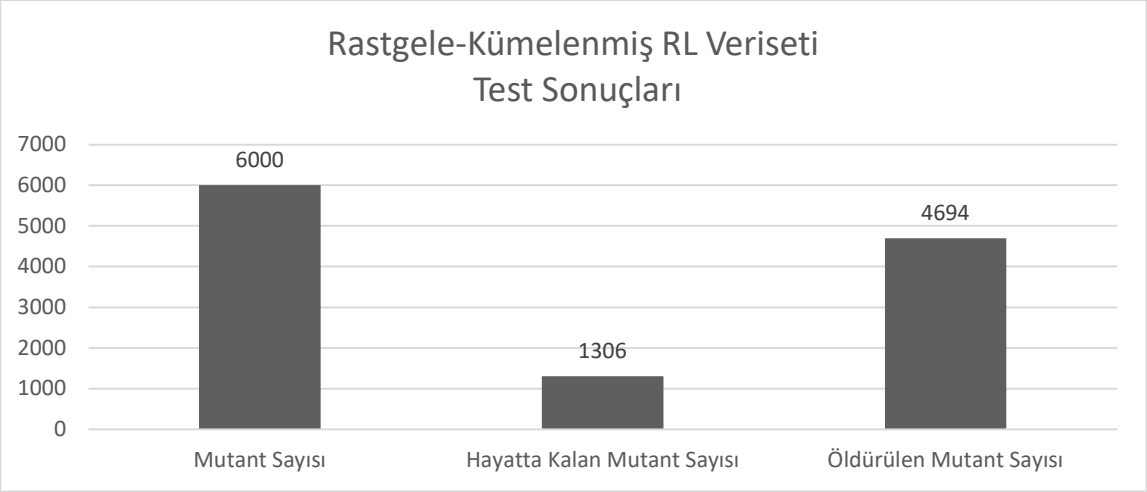


Şekil 4.4. Rastgele RL mutasyon testi sonuçları

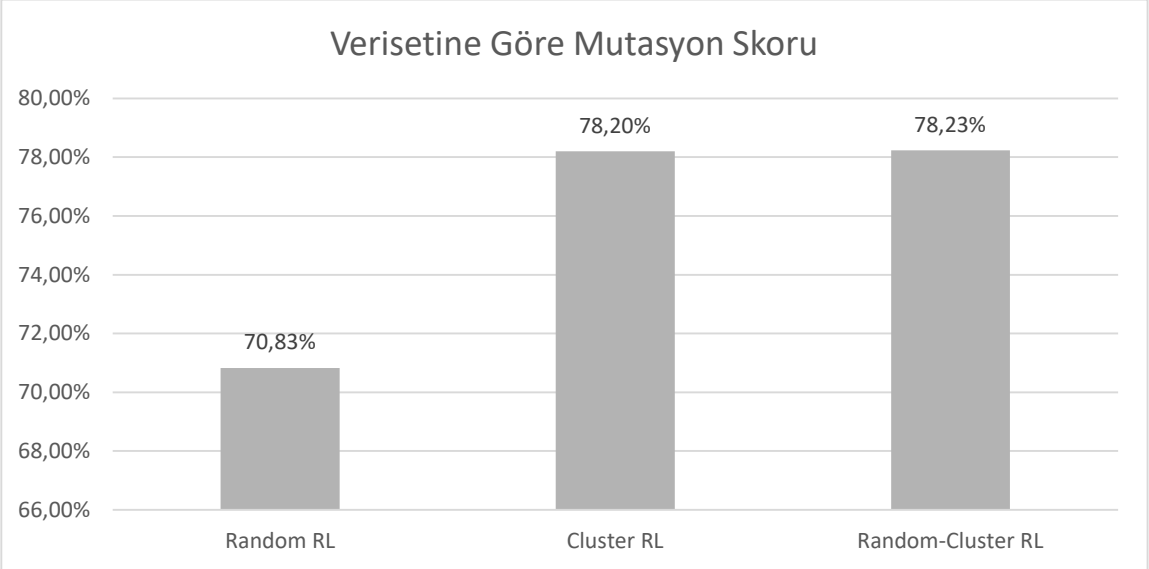


Şekil 4.5. Kümelenmiş RL mutasyon testi sonuçları

Veri kümelerine göre mutasyon puanı sonuçları: Rastgele Küme RL için %78,23, Kümelenmiş RL için %78,20 ve Rastgele RL için %70,83'tür (Şekil 4.7). Kümelenmiş veri seti için test sonuçları, modelin Rasgele veri seti kullanımından daha emniyetli olduğunu göstermektedir. Rastgele (Şekil 4.4), Kümelenmiş (Şekil 4.5) ve Rastgele Kümelenmiş (Şekil 4.6) test senaryolarının mutasyon test sonuçları için öldürülen mutasyon sayısının yüksek olması modellerin iyi yapılandırıldığını gösterse de hayatta kalan mutant ve test sonuçlarına göre güncellenip daha emniyetli hale getirilmesi önerilmektedir.



Şekil 4.6. Rastgele Kümelenmiş RL testi mutasyon testi sonuçları



Şekil 4.7. Veri Setlerine göre mutasyon skoru

Çizelge 4.3'te 3 ayrı veri setinde 5,10,20,40 müşterili konfigürasyonların testlerinin mutasyon skorları sunulmuştur. Çizelge 4.3'tede görüldüğü gibi veri setlerinin 5 müşterili versiyonlarında mutasyon skorlarından tam puan alırken problem karmaşıklaştıkça yani müşteri sayısı arttıkça performans düşüşleri yani mutasyon skorlarında düşüşler görülmektir. Bu durumda modelin karmaşık işlemlerde yetersiz kalabileceğini ön görülmektedir. Emniyet açısından geliştiricilerin test sonuçlarını inceleyerek modelin başarım sorunlarını çözmesi için kritik bilgi test sonuçlarıyla sağlanmaktadır.

Çizelge 4.3 RL testleri veri seti ve epizot sayısına göre mutasyon skorları

Epizot	**C5**	**C10**	**C20**	**C40**	**R5**	**R10**	**R20**	**R40**	**RC5**	**RC 10**	**RC 20**	**RC 40**
1	96,8	94,4	80	95,2	97,6	97,6	98,4	88,8	94,4	96	86,4	93,6
100	100	76,8	87,2	48,8	100	70,4	24	69,6	100	56	84,8	48
1000	100	97,6	76	17,6	100	36,8	72	74,4	100	100	70,4	84,8
2500	100	97,6	83,2	46,4	100	42,4	76,8	73,6	100	100	60	75,2
5000	100	94,4	60	64	100	68,8	60	72	100	100	52,8	44
10000	100	90,4	44	26,4	100	29,6	20,8	27,2	100	40,7	53,6	36

4.3. Dönüştürücü Tabanlı Model Testi

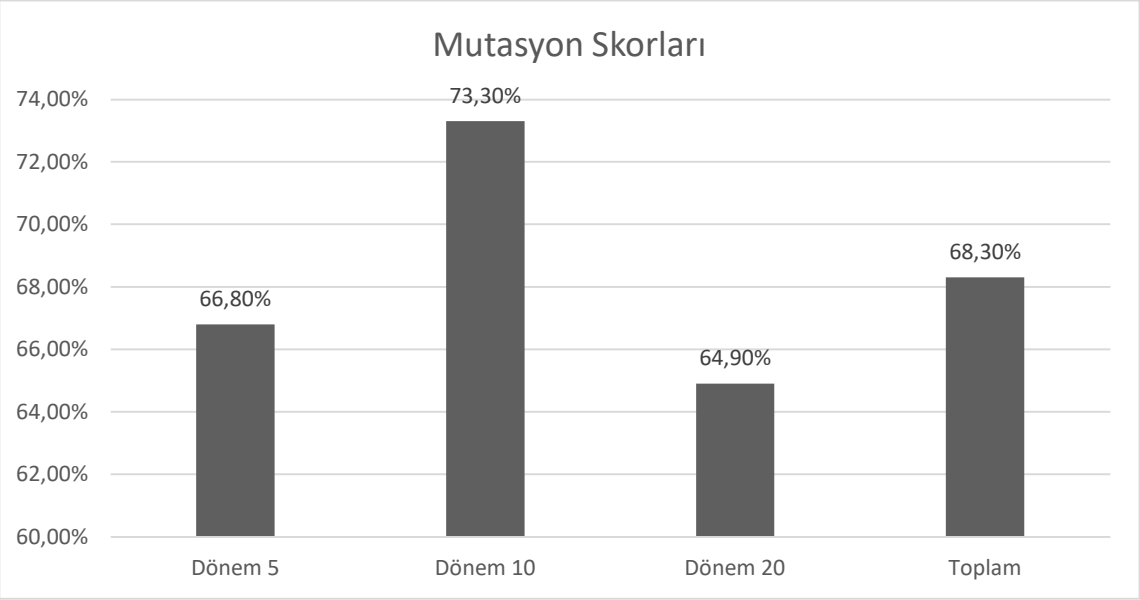
Dönüştürücü (Transformer) tabanlı model testleri, "TensorFlow" kütüphanesindeki Reuters veri kümesini kullanarak Dönüştürücü (Transformer) tabanlı bir model üzerinde gerçekleştirildi (Chollet vd, 2024; TensorFlow Authors, 2024). Gerçekleştirilen testlerde kullanılan hedef parametre ve mutasyon örnekleri Çizelge 4.4'te verilmiştir. Hedef parametre örneği Çizelge 4.4(a)'da verilmiştir. Kayıp (loss) fonksiyonunun ve Çizelge 4.4(c)'deki "maxlen" parametresinin mutasyonundan kaynaklanan örnek mutasyon Çizelge 4.4(b)'de verilmiştir. Model üç değerde test edildi: 5, 10 ve 20 dönem (epoch). Testlerde Çizelge 4.5'te görüldüğü gibi her testte 202 toplam 606 mutant üretilmiştir.

Çizelge 4.4 Dönüştürücü modeli mutasyon örneği

a) Transformers Model Hedef Parametre Örneği
maxlen=200 loss="sparse_categorical_crossentropy"
b) Mutasyona Uğratılmış Parametre Örneği
loss="binary_crossentropy"
c) Mutasyona Uğratılmış Parametre Örneği
maxlen=256

Çizelge 4.5 Sınıflandırmaya göre mutasyon testi sonuçları

Testler	**Öldürülen**	**Hayatta Kalan**	**Toplam**
Dönem 5	135	67	202
Dönem 10	148	54	202
Dönem 20	131	71	202
Toplam	414	192	606

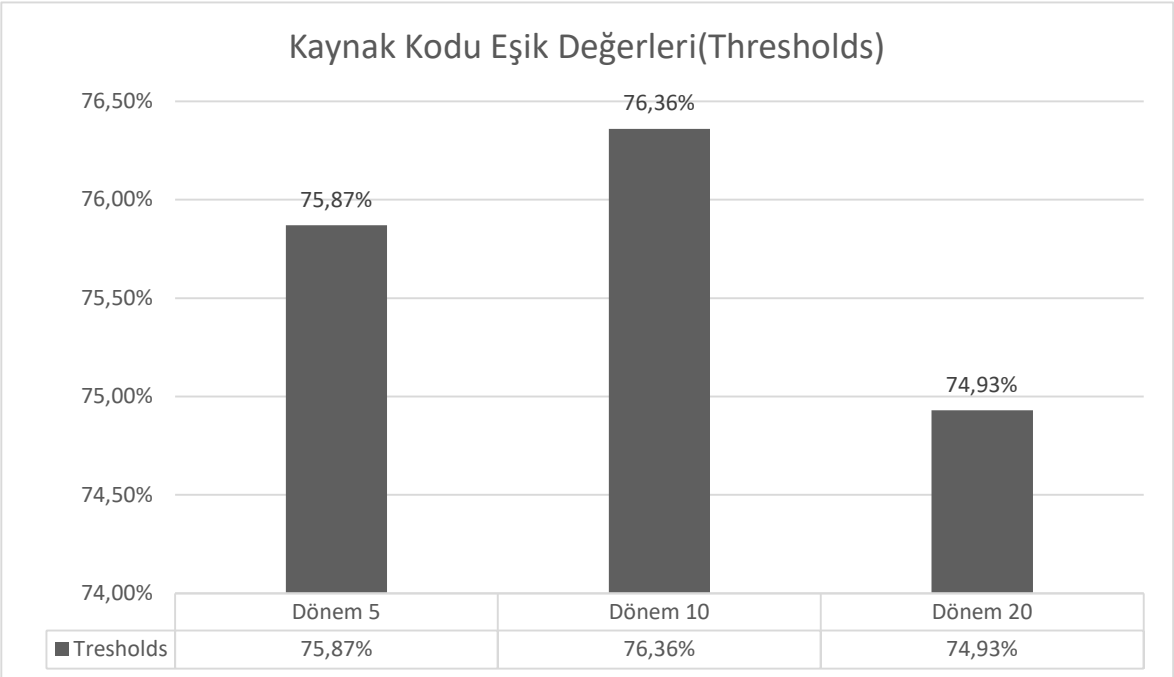


Şekil 4.8 Döneme göre mutasyon skorları

Şekil 4.8'da görüldüğü gibi 5, 10 ve 20 dönem (epoch) olarak üç farklı test sonucunda mutasyon skorları sırasıyla %66,8, %73,3 ve %64,9 olarak bulunmuştur. Genel mutasyon skoru ise 68.3% olarak sonuçlanmıştır.

Doğruluk (accuracy) skoru, testlerde oluşturulan mutantların sınıflandırılmasında eşik değeri olarak kullanıldığından, test sonuçlarını etkileyen ana faktörlerden biridir. Eşik değeri hedef modelin başarı metriğinin performans sonucuna göre belirlenmektedir. Testlerde eşik değerleri, Şekil 4.9'de görüldüğü gibi sırasıyla 5, 10 ve 20 dönem için %75,87, %76,36 ve %74,93'tür. Testler sonucunda birçok mutant hayatta kaldı olarak sınıflandırılmıştır. Bu sonuçlar, modülün test edilen modelinin zayıflıklarını ortaya koymaktadır. Modelin başarımı ve emniyeti, modelin performansı için daha iyi olan ortaya çıkan mutantların parametre seçimleri kullanılarak artırılabilir. Testler sonucunda en iyi mutasyon skoru 10 dönem olan testte ulaşılmıştır.

Hedef model doğruluk değerinden alınan eşik değeri %75,87 iken, dönem 10 testindeki 27. mutant %77,43 ile en yüksek doğruluğa sahiptir. Bu mutantta "units" nöron sayısı 20'den 90'a mutasyona uğramıştır. Bu sonuca göre, kaynak kodunun doğruluk değerinden %1,56 daha yüksek değere sahip bir mutant üretilmiştir. Bu tür performans artışları, test edilen modelin geliştirme sürecinin devam ederek iyileştirmesi gerekildiğini göstermektedir. Testler sonucunda DMM, modelleri başarılı bir şekilde test ederek araştırma ve geliştirme süreçlerinde modellerin kritik test ihtiyacının karşıladığını kanıtlamıştır.



Şekil 4.9 Hedef modelin eşik değerleri

4.4. Tartışma

LSTM, RL ve Dönüştürücü (Transformer) testlerinde üretilen mutantlar, modellerin mutasyonlara karşı verdiği tepkiler sonucunda başarım ve emniyet seviyeleri ortaya çıkmaktadır. Mutantların, hedef modele göre üstün başarım artışları, modellerin testler ışığında iyileştirilme yapılarak daha emniyetli ve yüksek başarımlı olabileceğini göstermektedir. Bir model düzgün çalışsa bile her zaman hata olasılığıyla karşı karşıyadır. Bu hataları en aza indirmenin tek yolu, modeli olabildiğince tutarlı sonuçlar vermesini sağlamaktır. Öte yandan modelin yüksek mutasyon skoru, daha iyi performans gösteren mutantlar üretemediği durumlarda ve büyük performans dalgalanmaları olmadığı durumlarda modelin kullanıma hazır olduğu değerlendirmesi yapılabilir. Uygulanan testler sonucunda 3 modelde de iyileştirmeye açık zayıf ve güçlü yönler ortaya çıkarılmıştır. Bu sayede, modellerin geliştirme süreçlerini yönlendirecek ve hızlandıracak kritik bilgiler geliştiricilere ve araştırmacılara sunulmuştur. DMM'in temel motivasyonu, geliştirilen sistemlerin model tabanlı yapılandırma doğruluğunu ve güvenliğini test etmektir. Testlerin genel amacı, modellerin uygulamalarda yüksek öğrenme kapasitesiyle ve minimum hata oranıyla çalışabilirliğini değerlendirmektir. Test sonuçları, modellerin öldürülen ve hayatta kalan mutantlarına ilişkin raporlar sunarak sürekli geliştirme sürecine katkı sağlar. DMM, SÜİT projesinde, modellerin geliştirilme sürecini destekleyen ve kullanıma hazır olma düzeylerini test eden kritik bir test aracı olarak görev yapmaktadır.

LSTM testlerinde, 5 dönemde 0.50, 10 dönemde 0.66, 20 dönemde 0.93 ve 30 dönemde 0.97 eşik değerleri gözlemlenmiştir. En iyi performansı gösteren mutantlar, 30 dönemde %2.2'lik bir performans artışı sağlayan Mutant 50 ve 51 olmuştur. Ayrıca, 10 dönemde Mutant 35, %17'lik bir performans artışı göstermiştir. Mutant 35'te uygulanan mutasyon, optimizasyon hiper parametresinin "adam"'dan "adagrad"'a değiştirilmesini içermektedir. Mutant 35'in, normalde 20 dönemde beklenen bir performans seviyesini sadece 10 dönem içinde göstermesi, modülün modeli en etkili bir şekilde test ettiğini ortaya koymaktadır.

RL testlerinde başarı metriği, "reward_max" olarak tanımlanmıştır. Bu metrik, elektrikli araçların çeşitli görevler sırasında izledikleri rota ve tükettikleri enerji gibi

faktörlerden etkilenir. Bu metriğin daha düşük bir değeri, daha iyi model performansını işaret eder. Rastgele Küme veri seti testinde gözlemlenen eşik değerler 1 epizot için 45926, 100 epizot için 35338, 1000 epizot için 35148, 2500 epizot için 33776, 5000 epizot için 33070 ve 10000 epizot için 33063 olarak kaydedilmiştir. En başarılı mutant, 10000 dönemde, “reward_max” değerini 3000 puan düşürerek 29000'e ulaşan Mutant 115 olmuştur. Bu mutantta uygulanan mutasyon, "gamma" parametresinin 0.99'dan 0.0001'e mutasyona uğratmasını içermektedir. “gamma” değerinin düşürülmesi, modelin kısa vadeli hedeflere yönelmesine yol açmış ve bu durumun performans artışına neden olabileceği, geliştiriciler tarafından bir yorum olarak dile getirilmiştir. Genel olarak, en başarılı mutantlar, “gamma” parametresinde yapılan mutasyonları içeren mutantlar olmuştur. Bu durum, test aracının model için uygun olmayan “gamma” değerlerinin tespit edilmesi ve geliştiricilerin hatalı ya da yetersiz yapılandırmaları iyileştirilmesini sağlamak için bilgi sunmuştur.

Dönüştürücü (Transformer) testlerinde, 5, 10 ve 20 dönem için eşik değerleri sırasıyla 75.87%, 76.36% ve 74.93% olarak gözlemlenmiştir. 5 dönemde en başarılı mutantlardan biri, 76.67% doğruluk performansı elde eden 200. mutant olmuştur. 10 dönemde ise en yüksek performansı 76.98% doğruluk ile 201. mutant göstermiştir. Her iki mutantta da "ff_dim" değerine ilişkin bir mutasyon söz konusudur: 200. mutantta 5 dönemde bu değer 32'den 50'ye, 201. mutantta ise 10 dönemde 32'den 64'e değiştirilmiştir. “ff_dim” değerindeki artış, modelin performans artışına olanak tanımıştır. DMM, hedef Dönüştürücü (Transformer) modelinin başarım ve emniyet seviyesini başarılı bir şekilde test etmiştir.

Bu çalışma kapsamında, farklı dönem veya epizot sayılarının artırılmasına yönelik geniş çaplı bir optimizasyon süreci yürütülmemiştir. Bunun temel nedeni, çalışmanın amacının model performansını artırmak değil, önceden optimize edilmiş modellerin test ve doğrulama süreçlerini değerlendirmektir. Yani, bu çalışma herhangi bir parametre optimizasyonu veya performans iyileştirme süreci değil; mutasyon tabanlı test teknikleriyle modellerin yapılandırma emniyetini, hata toleransı ve öğrenme davranışlarının değerlendirilmesi üzerine odaklanmıştır. Mutasyon testinin temel amacı, mevcut yapılandırmanın ne derece emniyetli ve doğru olduğunu ortaya koymaktır. Bu nedenle, modelin daha iyi hale getirilmesi veya parametre değerlerinin iyileştirilmesi gibi bir hedef güdülmemiş; yalnızca mevcut hâliyle sistemin davranışı, mutasyonlar karşısındaki tepkisi

ve potansiyel zayıflıkları değerlendirilmiştir. Bu yaklaşım, yazılım testlerinin doğrulama (verification) ilkesiyle örtüşmektedir: Sistem, belirli koşullar altında doğru ve kararlı bir şekilde çalışıyor mu, sorusuna odaklanılmaktadır. Bununla birlikte, yapılan testler sonucunda elde edilen çıktılar, geliştiricilerin modelin zayıf yönlerini, olası yapılandırma hatalarını ve iyileştirme potansiyeli taşıyan parametreleri tespit etmelerini sağlamaktadır. Bu veriler ışığında, geliştiriciler gerekli gördükleri durumlarda model üzerinde iyileştirmeler gerçekleştirebilmektedir. DMM test kapsamı sadece DNN yapısı ile sınırlı değil DNN tabanlı modellerin genel yapısını kapsamaktadır.

5.SONUÇ VE ÖNERİLER

Bu tez çalışmasında, Derin Sinir Ağları (DNN) tabanlı modellerin yüksek başarım, düşük hata oranı ve emniyet düzeyi açısından test edilmesi hedeflenmiştir. Bu doğrultuda geliştirilen Derin Mutasyon Modülü (DMM), yalnızca SÜİT projesi kapsamında değil, aynı zamanda farklı alanlarda kullanılan tüm DNN temelli sistemlerin doğrulama süreçlerine entegre edilebilecek modüler bir test altyapısı olarak tasarlanmıştır. Bu süreçte, model eğitiminde kullanılan parametrelere mutasyonlar uygulanmış ve modellerin mutasyonlara verdikleri tepkiler analiz edilmiştir. Sonuçlar, modellerin sorunlarını ortaya çıkarmanın yanı sıra, bu tür mutasyonlarla modellerin iyileştirilebilmesi için bir temel sunmuştur. İlgili alanlarda kullanılan modelleri daha emniyetli ve başarımlı sağlanabilmesi için test stratejileri ve metodolojileri sunulmuştur.

DMM GitHub'da (ESOGU-SRLAB, 2024) açık kaynak olarak mevcuttur, böylece dünya çapındaki araştırmacılar ve uygulayıcılar tarafından incelenebilir, test edilebilir ve geliştirilebilir. Bu tez çalışmasının sonuçları hem akademik literatüre hem de açık kaynak yazılım topluluğuna anlamlı katkılar sunmaktadır. Akademik açıdan, geliştirilen yaklaşım IEEE Access dergisinde yayımlanan "Mutation based white box testing of deep neural networks" başlıklı makalesi (Çetiner vd., 2024a) ve Akıllı Sistemlerde Yenilikler ve Uygulamaları (ASYU) Konferansı kapsamında sunulan "White box testing module for evaluation of transformers-based DNNs" bildirisi (Çetiner vd., 2024b) aracılığıyla bilimsel çevreye kazandırılmıştır. Açık kaynak olması, araştırmacı ve geliştiricilerin DMM'yi sürekli olarak iyileştirmesine ve yeni sorunlara uygulanmasına olanak tanımaktadır.

Bu tez çalışmasında önerilen Derin Mutasyon Modülü, literatürdeki DNN tabanlı modellerin test edilmesindeki zorluklara etkili bir çözüm olarak sunulmuştur. DMM, model parametrelerine özgü, sistematik ve kategorik mutasyonlar uygulayarak hata tespiti, yapılandırma doğruluğu ve performans değerlerinin değerlendirmesine imkân sağlamaktadır. Yapılan testler DMM'nin farklı mimarilere sahip modelleri mutasyon testi ile başarıyla test edebildiğini göstermektedir. Ayrıca DMM, özelleştirilebilir testler sayesinde esnek ve test maliyetini gözetmektedir. Aynı zamanda, DMM'nin parametrik test yaklaşımı sayesinde, ölçeklenebilir ve tekrarlanabilir bir test altyapısı sunulmaktadır. Bu tez

çalışmasında sunulan yöntem, ileride geliştirilecek test yaklaşımlarına da zemin oluşturabilecek niteliktedir.

Gelecekteki çalışmalarda, DMM'nin LLM mimarilerindeki etkinliğinin ve çeşitli uygulama alanlarına uygunluğunun değerlendirilmesi hedeflenmektedir. Bu doğrultuda, DMM'nin uygulanabilirliğinin ve kapsamının genişletilerek LLM'lerin test ihtiyaçlarının karşılanması amaçlanmaktadır. Ayrıca, daha özelleştirilebilir ve ölçeklenebilir test prosedürlerinin geliştirilmesine yönelik araştırmaların yürütülmesi planlanmaktadır. Diğer yandan, LLM gibi büyük ölçekli modeller için doğrulama ve onaylama süreçlerinin verimliliğinin artırılması da araştırmaların odak noktalarından biri olması amaçlanmaktadır.

KAYNAKLAR DİZİNİ

Ahuja, M. K., Gotlieb, A., & Spieker, H. (2022). Testing Deep Learning Models: A first comparative study of multiple testing techniques. In *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (pp. 130–137). https://doi.org/10.1109/ICSTW55395.2022.00035

Akay, B., Karaboga, D., & Sahin, O. (2021). Archive-based multi-criteria artificial bee colony algorithm for whole test suite generation. *Engineering Science and Technology, an International Journal, 24*(3), 806–817. https://doi.org/10.1016/j.jestch.2020.12.011

Arasteh, B., Imanzadeh, P., Arasteh, K., Gharehchopogh, F. S., & Zarei, B. (2022). A source-code aware method for software mutation testing using artificial bee colony algorithm. *Journal of Electronic Testing, 38*(3), 289–302.

Aslan Yıldız, Ö., Sarıçiçek, İ., & Yazıcı, A. (2025). *ESOGU-CEVRPTW* (Version 1) [Data set]. Mendeley Data. https://doi.org/10.17632/7vjzvxh72d.1

Ayubi, A., Taskin, F., Caglar, O., Asik, S., Baglum, C., & Yayan, U. (2023). Evaluation of deep neural network quality by CleanAI coverage metrics library. In *Proceedings of the International Conference on Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1–6). https://doi.org/10.1109/INISTA59065.2023.10310324

Baeza-Yates, R., & Liaghat, Z. (2017). Quality-efficiency trade-offs in machine learning for text processing. In *Proceedings of the IEEE International Conference on Big Data* (pp. 897–904).

Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2015). The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering, 41*(5), 507–525.

Braiek, H. B., & Khomh, F. (2020). On testing machine learning programs. *Journal of Systems and Software, 164*, 110542. https://doi.org/10.1016/j.jss.2020.110542

Çetiner, G., Yayan, U., & Yazıcı, A. (2024a). Mutation based white box testing of deep neural networks. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3482114

Çetiner, G., Bağlum, C., Yayan, U. ve Yazıcı, A. (2024b). *White box testing module for evaluation of transformers-based DNNs*. 2024 Innovations in Intelligent Systems and Applications Conference (ASYU), Ankara, Türkiye, ss. 1-6. https://doi.org/10.1109/ASYU62119.2024.10757012

Chollet, F., & the TensorFlow Authors. (2024). Reuters Newswire Classification Dataset. *Keras Documentation*. https://keras.io/api/datasets/reuters/

KAYNAKLAR DİZİNİ

Davis, M. D., & Weyuker, E. J. (1981). Pseudo-oracles for non-testable programs. In *Proceedings of the ACM '81 Conference* (pp. 254–257).

DLR. (2025). Simulation of Urban MObility (SUMO). https://sumo.dlr.de

Ding, J., Kang, X., & Hu, X.-H. (2017). Validating a deep learning framework by metamorphic testing. In 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET) (pp. 28–34). https://doi.org/10.1109/MET.2017.2

ESOGU-SRLAB. (2024). DNN-Mutator. *GitHub repository*. https://github.com/ESOGU-SRLAB/DNN-Mutator

Ferreira, F., Silva, L. L., & Valente, M. T. (2021). Software engineering meets deep learning: A mapping study. In Proceedings of the ACM Symposium on Applied Computing (pp. 1542–1549). https://doi.org/10.1145/3412841.3442029

Felderer, M., & Ramler, R. (2021). Quality assurance for AI-based systems: Overview and challenges. In *Proceedings of the 32nd International Conference on Software Engineering and Knowledge Engineering* (pp. 3–12). Springer. https://doi.org/10.1007/978-3-030-65854-0_3

FIWARE Foundation. (2023). FIWARE: The open-source platform for our smart digital future. https://www.fiware.org

Guesmi, A., Hanif, M. A., Ouni, B., & Shafique, M. (2023). Physical adversarial attacks for camera-based smart systems: Current trends, categorization, applications, research challenges, and future outlook. *IEEE Access, 11*, 109617–109668. https://doi.org/10.1109/ACCESS.2023.3321118

Henard, C., Papadakis, M., Harman, M., Jia, Y., & Le Traon, Y. (2016). Comparing white-box and black-box test prioritization. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (pp. 523–534). https://doi.org/10.1145/2884781.2884791

Hu, Q., Ma, L., Xie, X., Yu, B., Liu, Y., & Zhao, J. (2019). DeepMutation++: A mutation testing framework for deep learning systems. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 1158–1161). https://doi.org/10.1109/ASE.2019.00126

Jia, Y., & Harman, M. (2011). An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering, 37*(5), 649–678. https://doi.org/10.1109/TSE.2010.62

KAYNAKLAR DİZİNİ

Klampfl, L., Chetouane, N., & Wotawa, F. (2020). Mutation testing for artificial neural networks: An empirical evaluation. In *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS)* (pp. 356–365). https://doi.org/10.1109/QRS51102.2020.00054

Kusharki, M. B., Misra, S., Muhammad-Bello, B., Salihu, I. A., & Suri, B. (2022). Automatic classification of equivalent mutants in mutation testing of Android applications. Symmetry, 14(4), 820.

Li, J., Li, S., Wu, J., Luo, L., Bai, Y., & Yu, H. (2022). MMOS: Multi-staged mutation operator scheduling for deep learning library testing. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)* (pp. 6103–6108). https://doi.org/10.1109/GLOBECOM48099.2022.10001093

Li, Z., Cui, Z., Liu, J., Zheng, L., & Liu, X. (2020). Testing neural network classifiers based on metamorphic relations. In *Proceedings of the 2019 6th International Conference on Dependable Systems and Their Applications (DSA)* (pp. 389–394). https://doi.org/10.1109/DSA.2019.00060

Liu, Q., et al. (2020, July). Monitoring the health of emerging neural network accelerators with cost-effective concurrent test. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)* (pp. 1–6).

Lu, Y., Shao, K., Sun, W., & Sun, M. (2022). RGChaser: A RL-guided fuzz and mutation testing framework for deep learning systems. In *Proceedings of the International Conference on Dependable Systems and Their Applications (DSA)* (pp. 12–23). https://doi.org/10.1109/DSA56465.2022.00012

Ma, L., et al. (2018). DeepMutation: Mutation testing of deep learning systems. In *Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)* (pp. 100–111). https://doi.org/10.1109/ISSRE.2018.00021

Marijan, D., & Gotlieb, A. (2020). Software testing for machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(9), 13576–13582. https://doi.org/10.1609/AAAI.V34I09.7084

Marijan, D., Gotlieb, A., & Ahuja, M. K. (2019). Challenges of testing machine learning based systems. In *Proceedings of the IEEE International Conference on Artificial Intelligence Testing (AITest)* (pp. 101–102). https://doi.org/10.1109/AITEST.2019.00010

KAYNAKLAR DİZİNİ

Moussa, D. A., Hefenbrock, M., & Tahoori, M. (2024). Testing for multiple faults in deep neural networks. *IEEE Design & Test, 41*(3), 47–53. https://doi.org/10.1109/MDAT.2024.3365988

Naeem, M. R., Lin, T., Naeem, H., Ullah, F., & Saeed, S. (2019). Scalable mutation testing using predictive analysis of deep learning model. *IEEE Access, 7*, 158264–158283. https://doi.org/10.1109/ACCESS.2019.2950171

Numan, G. (2020). Testing artificial intelligence. In S. Goericke (Ed.), *The future of software quality assurance* (pp. 169–185). Springer. https://doi.org/10.1007/978-3-030-29509-7_10

Özdemir, Ö., & Demir, D. (2021). TEST-INT: A testing platform for deep learning models. In *Proceedings of the Turkish National Software Engineering Symposium (UYMS)* (pp. 1–3). https://doi.org/10.1109/UYMS54260.2021.9659790

Panichella, A., & Liem, C. C. L. (2021). What are we really testing in mutation testing for machine learning? A critical reflection. In *Proceedings of the IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (pp. 66–70). https://doi.org/10.1109/ICSE-NIER52604.2021.00022

Petrović, G., Ivanković, M., Fraser, G., & Just, R. (2022). Practical mutation testing at scale: A view from Google. *IEEE Transactions on Software Engineering, 48*(10), 3900–3912. https://doi.org/10.1109/TSE.2021.3107634

Raju. (2021). Mutation testing framework for machine learning. *arXiv*. https://doi.org/10.48550/arXiv.2102.10961

Ramanathan, A., Pullum, L. L., Hussain, F., Chakrabarty, D., & Jha, S. K. (2016). Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition* (pp. 786–791).

Sbai, Z. (2025). Model checking deep neural networks: Opportunities and challenges. *Frontiers in Computer Science, 7*. https://doi.org/10.3389/fcomp.2025.1557977

Shen, W., Wan, J., & Chen, Z. (2018). MuNN: Mutation analysis of neural networks. In *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 108–115). https://doi.org/10.1109/QRS-C.2018.00032

KAYNAKLAR DİZİNİ

SÜİT – Sürdürülebilir Kentler için İleri Teknolojiler Platformu. (2023). *Sürdürülebilir Kentler için İleri Teknolojiler Platformu – Ana Sayfa*. Erişim: https://suit.metu.edu.tr/, Erişim tarihi: 16.04.2025.

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE, 105*(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

TensorFlow Authors. (2024). TensorFlow models: Transformer. *GitHub repository*. https://github.com/tensorflow/models/tree/master/official/legacy/transformer

Wei, J., Fan, M., Jiao, W., Jin, W., & Liu, T. (2024). BDMMT: Backdoor sample detection for language models through model mutation. *IEEE Transactions on Information Forensics and Security*. https://doi.org/10.1109/TIFS.2024.3376968

Wei, Y., Huang, S., Wang, Y., Liu, R., & Xia, C. (2022). Mutation testing based safety testing and improving on DNNs. In *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS)* (pp. 821–829). https://doi.org/10.1109/QRS57517.2022.00087

Weyuker, E. J. (1982). On testing non-testable programs. *The Computer Journal, 25*(4), 465–470.

Wu, H., Li, Z., Cui, Z., & Zhang, J. (2021). A mutation-based approach to repair deep neural network models. In *Proceedings of the International Conference on Dependable Systems and Their Applications (DSA)* (pp. 730–731). https://doi.org/10.1109/DSA52907.2021.00106

Yayan, U., & Aşik, S. (2023a). Generating Python mutants from bug fixes using neural machine translation. *IEEE Access, 11*, 85678–85693. https://doi.org/10.1109/ACCESS.2023.3302695

Yayan, U., & Baglum, C. (2023b). Tailored mutation-based software fault injection tool (IM-FIT). *SoftwareX, 23*, 101463. https://doi.org/10.1016/j.softx.2023.101463

Yılmaz, M., Çinar, E., & Yazıcı, A. (2025). A transformer-based model for state of charge estimation of electric vehicle batteries. *IEEE Access, 13*, 33035–33048. https://doi.org/10.1109/ACCESS.2025.3542961

Yu, J., Duan, S., & Ye, X. (2023). A white-box testing for deep neural networks based on neuron coverage. *IEEE Transactions on Neural Networks and Learning Systems, 34*(11), 9185–9197. https://doi.org/10.1109/TNNLS.2022.3156620

KAYNAKLAR DİZİNİ

Zhang, J., et al. (2016). Predictive mutation testing. In *Proceedings of the 25th International Symposium on Software Testing and Analysis* (pp. 342–353).

Zhang, J., Zhang, L., Hao, D., Wang, M., & Zhang, L. (2019a). Do pseudo test suites lead to inflated correlation in measuring test effectiveness? In *Proceedings of the 12th IEEE Conference on Software Testing, Validation and Verification* (pp. 252–263).

Zhang, X., Yin, Z., Feng, Y., Shi, Q., Liu, J., & Chen, Z. (2019b). NeuralVis: Visualizing and interpreting deep learning models. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 1106–1109). https://doi.org/10.1109/ASE.2019.00113