T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTO-KOCLAYICI MİMARISI KULLANARAK MERMER YÜZEY ANOMALİ TESPİTİ

YÜKSEK LİSANS TEZİ

Muhammad Yahya ABDULLAH

Bilgisayar Mühendisliği Anabilim Dalı

OCAK 2024

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTO-KOCLAYICI MİMARISI KULLANARAK MERMER YÜZEY ANOMALİ TESPİTİ

YÜKSEK LİSANS TEZİ

Muhammad Yahya ABDULLAH

Bilgisayar Mühendisliği Anabilim Dalı

Prof. Dr. Cemil ÖZ

OCAK 2024

Muhammad Yahya ABDULLAH tarafından hazırlanan "Otomatik Kodlayıcı Mimarisi Kullanarak Mermer Yüzey Anomali Tespipi" adlı tez çalışması 25.01.2024 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri EnstitüsüBilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı : **Prof. Dr. Cemil Öz**
Sakarya Üniversitesi

Jüri Üyesi : **Doç. Dr. Nilüfer Yurtay**
Sakarya Üniversitesi

Jüri Üyesi : **Dr. Öğretim Üyesi Selman Hızal**
Sakarya Uygulamalı Bilimier Üniversitesi

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİX

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "OTO-KODLAYICI MİMARISI KULLANARAK MERMER YÜZEY ANOMALİ TESPİTİ" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, etik kurul onay belgesi aldığımı (etik onayı gerekmiyorsa bu cümle metinden çıkartılır), çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(25/01/2024).

Muhammad Yahya ABDULLAH

Canım Eşime, Aileme ve Bütün Eğitmenlerime …

TEŞEKKÜR

Bu tezin yazımında bana zaman ayırarak yol gösteren, beni yönlendiren, değerli geri bildirimler sağlayan ve tüm aşamalarında yardımlarını esirgemeyen değerli danışman hocam Prof. Dr. Cemil Öz'e teşekkürlerimi sunarım. Yine de bu tezin hazırlanması sırasında bana zaman ayırdığı, yol gösterdiği ve değerli bilgiler verdiği için hocam Dr. Muhammed Ali Nur Öz'e teşekkür ederim. Ayrıca, bu tezi tamamlamam için bana motivasyon ve güç sağlayan sevgili eşim Tsania Kareema'ya teşekkürlerimi sunarım.

Muhammad Yahya Abdullah

İÇİNDEKİLER

Sayfa

KISALTMALAR

2D : Two Dimensional (İki Boyutlu)
3D : Three Dimensional (Üç Boyutlu)
ANN : Artificial Neural Network (Yapay Sınır Ağı)
CNN : Convolutional Neural Network (Evrişimsel Sinir Ağı)
GAN : Generative Adversarial Network (Çekişmeli Üretici Ağlar)
GPU : Graphics Processing Unit (Grafik İşlem Birimi)
GUI : Graphical User Interface (Grafik Kullanıcı Arayüzü)
IDE : Integrated Development Environment (Entegre Geliştirme Ortamı)
IDS : Intrusion Detection Systems (Saldırı Tespit Sistemleri)
IoT : Internet of Things (Nesnelerin İnterneti)
JPEG : Joint Photographic Experts Group (Ortak Fotoğraf Uzmanları Grubu)
KL : Kullback-Leibler
ML : Mechine Learning (Mekanik Öğrenme)
MRI : Magnetic Resonance Imaging (Manyetik Rezonans Görüntüleme)
NMT : Neural Machine Translation (Nöral Makine Çevirisi)
NN : Neural Network (Sinir Ağı)
PNN : Probabilistic Neural Network (Olasılıksal Sinir Ağı)
RAM : Random Access Memory (Rastgele Erişimli Bellek)
RNN : Recurrent neural network (Tekrarlayan sinir ağı)
URL : Uniform Resource Locator (Tekdüzen Kaynak Konum Belirleyici)

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

OTO-KODLAYICI MİMARİSİ KULLANARAK MERMER YÜZEY ANOMALİ TESPİTİ

ÖZET

Mermer, yer kabuğundaki yüksek basınç ve ısı nedeniyle yapısal ve kimyasal dönüşümler geçiren kireçtaşından oluşan bir tür metamorfik kayadır. Mermer pürüzsüz bir dokuya sahiptir ve genellikle güzel desenler veya damarlar içerir. Taş, geldiği kayanın içerdiği minerallere bağlı olarak genellikle beyaz, gri, pembe ve yeşil gibi çeşitli renklerde olur. Mermer, sanat, mimari ve dekorasyon alanlarında oldukça değerli olmasını sağlayan benzersiz bir estetik ve zarafete sahiptir. Zeminler, duvarlar, heykeller, masalar ve tezgahlar dahil olmak üzere çeşitli uygulamalar için yaygın olarak kullanılmaktadır. Nispeten yüksek sertliği, aşınmaya karşı dirençli ve dayanıklı olmasını sağlar.

Çeşitli uygulamalarda ve endüstrilerde kullanımı her yıl ilerledikçe, mermer kullanıcılarının sayısı, çeşitli bölgelerdeki eğilimlere ve ekonomik koşullara ve özel projelerle ilgili talebe bağlı olarak dalgalanmaya devam edecektir. Birçok tüketici mermeri güzel desenleri ve uzun ömürlü dayanıklılığı için satın aldığından, mermer fabrikalarındaki yüzey hatalarının izlenmesi ve mermerin kalitesinin korunması, tüketici güvenini artırmak için çok önemlidir. Ayrıca hatalı üretimden kaynaklanan sorunların erken önlenmesi, aksi takdirde oluşabilecek maddi kayıpların da önüne geçmektedir. Bu nedenle imalat sanayinde yüzey hatalarının tespit edilmesi çok önemlidir.

Geleneksel yüzey hatası denetimi genellikle insanların görsel denetimini veya mikroskop gibi basit ekipmanların kullanımını içerir. Üretim bağlamında, çizikler, çatlaklar veya yüzey düzgünsüzlüğü gibi kusurları tespit etmek için operatörler tarafından manuel denetim hala yaygın olarak kullanılmaktadır. Bu yöntemler etkili olabilmekle birlikte, zaman alıcı olma eğilimindedir, operatör uzmanlığına bağlıdır ve tutarlı bir doğruluk düzeyi sağlamayabilir.

Otomatik kontrol sistemleri, yüzey kusurları denetiminin verimliliğini ve doğruluğunu artırmada ilerici bir çözüm haline gelmektedir. Otomatik kontrol sistemleri görüntü tanıma, yapay zeka ve makine görüşü gibi teknolojileri entegre ederek yüzey kusurlarını yüksek doğrulukla otomatik olarak tespit edebilir ve sınıflandırabilir. Bu da insan faktörüne olan bağımlılığı azaltır ve üretim ortamlarında verimliliği artırır.

Daha önceki bazı çalışmalarda, Sonuç Ağırlıklandırma tabanlı Resnet Özellik Piramidi Ağı (SA-ROPA), Evrişimsel Sinir Ağı (CNN), Üretken Adversial Ağ (GAN), Segmentasyon tabanlı derin öğrenme yüzey kusurlarının tespitinde (malzeme veya diğer görevler) uygulanmıştır. Bu araştırmada, bir tür yapay sinir ağı Autoencoders mimarisi, Keras kütüphaneleri ve Python programlama dili kullanarak mermer yüzeylerindeki anormallikleri tespit etmeyi amaçlayan bir algoritma oluşturduk.

Otomatik kodlayıcı, veri temsilini otomatik olarak öğrenen bir sinir ağı türüdür ve bu bağlamda normal verileri yüksek doğrulukla yeniden yapılandırmak için kullanılır. Model çok sayıda normal veri örneği gördüğünde, iyi bir yeniden yapılandırma

üretebilir. Ayrıca otoenkoder modeli, modelin verileri ne kadar iyi yeniden yapılandırdığını ölçmek için bir kayıp fonksiyonu da kullanır. Model anormal verilerle karşılaştığında, kayıp fonksiyonu olağandışı verilerin yeniden yapılandırılmasındaki zorluk nedeniyle artma eğilimindedir. Daha sonra kayıp fonksiyonunun değeri kullanılarak bir eşik değeri belirlenebilir. Önceden tanımlanmış eşik değerini aşan bir kayıp değerine sahip veriler anormallik olarak kabul edilebilir. Bu, otomatik kodlayıcının normal verilere karşılık gelmeyen örüntüleri otomatik olarak tespit etmesini sağlar.

Bu çalışmada iyi sonuçlar elde etmek için aynı veri kümesini kullanarak ancak farklı otoenkoder mimarisi modelleriyle ve farklı eşik değerleri alarak birkaç deneme yaptık. ilk modelde, kodlayıcı bölümünde MaxPooling2D kullanarak dört evrişim katmanı ve dört havuzlama katmanı ve ayrıca kod çözücü bölümünde UpSampling2D kullanarak dört evrişim katmanı ve dört havuzlama katmanı içeren bir otoenkoder mimarisi oluşturduk. Daha sonra oluşturulan modeli eğitmek için eğitim verisi olarak 300 görüntü verisi ve doğrulama verisi olarak 50 görüntü verisi sağlıyoruz. Ayrıca optimizer olarak Adam, 50 adet bacth boyutu ve kayıp fonksiyonu 'mean_squared_error' şeklinde hiper parametreler de kullanılmaktadır. daha sonra eğitilen model ile yeniden yapılandırma hatasının ortalama değerini ve giriş verilerinin standart sapma değerini bulmak için bir fonksiyon yapılır ve iki değer toplanır ve daha sonra anormallikleri tespit etmede bir parametre olarak eşik olarak kullanılır.

Daha sonra ikinci model için kodlayıcı bölümünde MaxPooling2D fonksiyonunu kullanarak üç konvolüsyon katmanı ve üç havuzlama katmanı içeren bir otoenkoder mimarisi oluşturduk ve ayrıca kod çözücü bölümünde UpSampling2D fonksiyonunu kullanarak üç konvolüsyon katmanı ve üç havuzlama katmanı oluşturduk. Bundan sonra, eğitilen model ile anormallikleri tespit etmek için daha önce oluşturulan fonksiyonu parametre olarak kullanarak eşik değerini arıyoruz. Daha sonra üçüncü deneyde, ikinci deneydeki aynı modeli kullandık, yani kodlayıcı bölümünde üç konvolüsyon katmanı ve üç havuzlama katmanı ve kod çözücü katmanı için üç konvolüsyon katmanı ve üç havuzlama katmanı yaptık. Ancak bu deneyde eşik değerini doğrudan eğitilmiş modelin ortalama yeniden yapılandırma hata değerinden aldık ve anormallikleri tespit etmede bir parametre olarak kullandık.

Parametreler elde edildikten sonra, veri kümesi sağlayıcı web sitelerinden birinden elde edilen "mermer yüzey anomali tespiti" veri kümesi kullanılarak mermer yüzeyindeki anomalileri tespit etmek için bir fonksiyon oluşturulmuştur. Elli adet normal mermer yüzey görüntüsü ve elli adet kusurlu (noktalar, eklemler, çatlaklar) mermer yüzey görüntüsü anormallik tespit fonksiyonunu test etmek için kullanılmıştır. İlk deney için doğruluk sonucu %74, ikinci deney için doğruluk sonucu %84 ve son olarak üçüncü deneyde sonuç %88'dir.

Bu çalışmanın sonuçları arasında bu yöntemin anormallikleri tespit etmedeki etkinliği, otoenkoder mimarisinin diğer yöntemlere kıyasla avantajları ve bu bulguların çeşitli alanlardaki potansiyel pratik uygulamaları yer almaktadır. Otomatik kodlayıcının verileri yeniden yapılandırma yeteneğinden yararlanılarak, yeniden yapılandırılan sonuçlar orijinal verilerle karşılaştırılarak anormallik tespiti gerçekleştirilebilir. Bu yaklaşımın temel avantajı, sınıf etiketlerine ihtiyaç duymadan anormallik tespiti sorununu ele alma kabiliyetinde yatmaktadır.

MARBLE SURFACE ANOMALY DETECTİON USİNG AUTOENCODER ARCHİTECTURE

SUMMARY

Marble is a type of metamorphic rock composed of limestone that has undergone structural and chemical transformations due to high pressure and heat in the earth's crust. Marble has a smooth texture and often contains beautiful patterns or veins. The stone usually comes in a variety of colors, such as white, gray, pink and green, depending on the minerals contained in the rock it comes from.

Marble has a unique aesthetic and elegance that makes it highly valued in art, architecture and decoration. It is widely used for a variety of applications, including floors, walls, sculptures, tables and countertops. Its relatively high hardness makes it wear resistant and durable.

As its use in various applications and industries advances each year, the number of marble users will continue to fluctuate depending on trends and economic conditions in various regions and demand for special projects.

As many consumers buy marble for its beautiful patterns and long-lasting durability, monitoring surface defects in marble factories and maintaining the quality of the marble is crucial to increase consumer confidence.

In addition, early prevention of problems caused by faulty production prevents financial losses that might otherwise occur. Therefore, it is very important to detect surface defects in the manufacturing industry.

Traditional surface defect inspection usually involves visual inspection by humans or the use of simple equipment such as microscopes. In a manufacturing context, manual inspection by operators to detect defects such as scratches, cracks or surface unevenness is still widely used. While these methods can be effective, they tend to be time-consuming, depend on operator expertise and may not provide a consistent level of accuracy.

Automated inspection systems are becoming a progressive solution to improve the efficiency and accuracy of surface defect inspection. By integrating technologies such as image recognition, artificial intelligence and machine vision, automated inspection systems can automatically detect and classify surface defects with high accuracy. This reduces reliance on the human factor and increases efficiency in production environments.

An autoencoder is a type of neural network that automatically learns the data representation and in this context is used to reconstruct normal data with high accuracy. When the model sees a large number of normal data samples, it can produce a good reconstruction. The autoencoder model also uses a loss function to measure how well the model reconstructs the data.

When the model encounters abnormal data, the loss function tends to increase due to the difficulty in reconstructing unusual data. A threshold value can then be set using the value of the loss function. Data with a loss value that exceeds the predefined threshold value can be considered anomalous. This allows the autoencoder to automatically detect patterns that do not correspond to normal data.

To get good results in this study we conducted several trials using the same dataset but with different autoencoder architecture models and taking different threshold values. in the first model we created an autoencoder architecture with four convolution layers and four pooling layers using MaxPooling2D in the encoder section and also four convolution layers and four pooling layers using UpSampling2D in the decoder section.

Then we provide 300 image data as training data and 50 image data as validation data to train the model that has been created. In addition, hyper parameters are also used in the form of Adam as an optimizer, 50 number of batch sizes and the loss function 'mean_squared_error'. then with the model that has been trained, a function is made to find the average value of the reconstruction error and the standard deviation value of the input data and the two values are summed up and then used as a threshold as a parameter in detecting anomalies.

Then for the second model we created an autoencoder architecture with three convolution layers and three pooling layers using the MaxPooling2D function in the encoder section and also in the decoder section we created three convolution layers and three pooling layers using the UpSampling2D function. After that, with the model that has been trained, we look for the threshold value using the previously created function as a parameter to detect anomalies.

Then in the third experiment we used the same model in the second experiment, namely three convolution layers and three pooling layers in the encoder section as well as for the decoder layer made three convolution layers and three pooling layers. But in this experiment we took the threshold value directly from the average reconstruction error value of the trained model and used it as a parameter in detecting anomalies.

After the parameters are obtained, a function is created to detect anomalies on the marble surface using the "marble surface anomaly detection" dataset obtained from one of the dataset provider websites. Fifty images of normal marble surfaces and fifty images of marble surfaces with defects (points, joints, cracks) were used to test the anomaly detection function. For the first experiment, the accuracy result is 74%, for the second experiment, the accuracy result is 84% and finally in the third experiment, the result is 88%.

The conclusions of this study include the effectiveness of this method in detecting anomalies, the advantages of the autoencoder architecture compared to other methods, and the potential practical applications of these findings in various fields. By utilizing the ability of the autoencoder to reconstruct data, anomaly detection can be performed by comparing the reconstructed results with the original data. The main advantage of

this approach lies in its ability to address the problem of anomaly detection without the need for class labels.

1. GİRİŞ

Bu çalışmada, bir tür sinir ağı olan otomatik kodlayıcı mimarisi kullanılarak mermer yüzeylerdeki anormallikleri tespit etmek için bir algoritma tasarlanmıştır. Bu çalışmada, önerilen modeli ve algoritmayı test etmek için "mermer yüzey anomalisi tespiti" veri kümesi kullanılmıştır. Bu araştırmada elde edilen son doğruluk sonucu 88%'dir. Kullandığımız veri kümesi beyaz bir mermer yüzeye sahiptir, bu nedenle şu anda oluşturulan algoritma sadece beyaz mermer yüzeylere uygulanabilir. Ancak gelecekte başka mermer veri kümeleri bularak ve bunları önerilen modelle eğiterek diğer renkli mermer türlerine de uygulanması mümkündür.

1.1. Tezin Arka Planı (Background)

Mermer, kireçtaşı ve çoğunlukla kalsit veya dolomit olmak üzere yeniden kristalleşmiş karbonat minerallerinden oluşan metamorfik bir kayadır. Mermer, güzelliği ve dayanıklılığı ile ödüllendirilen "zamansız bir malzeme" olarak da adlandırılır [1]. Mermer çok yönlü ve dayanıklı bir malzemedir, çeşitli uygulamalarda ve sektörlerde kullanımı her yıl devam etmektedir, mermer kullanıcılarının sayısı da çeşitli bölgelerdeki eğilimlere ve ekonomik koşullara ve özel projelerle ilgili talebe bağlı olarak dalgalanmaya devam edecektir.

Şekil 1.1. bölgeye göre mermer kullanici büyüme tahmini

Birçok tüketici mermeri güzel desenleri ve uzun dayanıklılığı nedeniyle satın aldığından, mermer fabrikalarının mermer yüzey hatalarının izlenmesini iyileştirmesi, tüketicinin fabrikadan mermer almaya devam etme güvenini korumak için çok önemlidir. Buna ek olarak, hatalı üretimden kaynaklanan sorunların erken önlenmesi, aksi takdirde meydana gelebilecek maddi kayıpları önler [2, 3]. Bu nedenlerden dolayı, imalat sanayinde yüzey hatalarının tespit edilmesi çok önemlidir.

Yüzey kusurları denetimi en önemli kalite kontrol bileşenlerinden biridir. Geleneksel yüzey kusur denetimi insan gözetimi altında manuel olarak yapılır [4, 5]. Bu yöntemin zaman kaybı, düşük verimlilik ve çalışan deneyimi ve motivasyonu gibi birçok faktöre bağlı olduğu için sübjektif sonuçlar gibi dezavantajları vardır [6]. Düşük kontrastlı yüzeylerde ortaya çıkan ince kusurlar, eğitimli denetçilerle bile net bir şekilde belirlenemez [7]. Bu nedenle, çalışma süresi, yorgunluk ve stresten etkilenmeyen bir otomasyon sisteminin geliştirilmesi bu işlemi objektif olarak gerçekleştirebilir [8, 9]. Görüntü işleme kullanan otomatik kontrol sistemleri, manuel kontrolün dezavantajlarının çoğunun üstesinden gelebilir ve üreticilere kaliteyi artırma ve maliyetleri önemli ölçüde azaltma fırsatı sunar [10].

Teknolojinin hızla gelişmesi ve endüstriye uygulanmasının gelişmesi sonucunda hızlı ve özel ekipmanlar geliştirilmiş ve yapay görme algoritmaları gerçek dünyadaki endüstriyel denetim problemlerine kolayca uygulanmıştır. Yapay görme, görüntü işleme, optik, örüntü tanıma, yapay zeka gibi birçok ileri teknolojiyi içerir. Ayrıca endüstriyel ürün kontrolünde düşük maliyet, yüksek hız, yüksek doğruluk, gerçek zamanlı performans ve yüksek kalite sağlar [11, 12].

Anomali tespit sistemlerinin karşılaştığı başlıca zorluklar aşağıdaki gibidir. Ezik, leke, çatlak, çizik, kir, çizik vb. gibi anormallikler, buna ek olarak, veriler genellikle gürültü de içerir, böylece verileri anormal hale getirir ve sistemin bunu bir anormallik olarak sınıflandırması nadir değildir. Aydınlatma, sıcaklık, aşırı hava koşulları (kar gibi) gibi çevresel faktörler de tespit sistemini etkiler [13, 14]. Bu zorluklar, dokulu yüzeylerde anomali tespiti görevini karmaşık ve zor hale getirmektedir.

Yüzey hatası tespitinde daha önce yapılan bazı çalışmalarda, Hüseyin Üzen ve iki arkadaşı Muammer Türkoğlu ve Davut Hanbay tarafından "Yüzey Hatası Tespiti için Resnet Öznitelik Piramit Ağı Mimarisi" başlıklı makale ile yürütülen Sonuç Ağırlıklandırma Tabanlı Resnet Öznitelik Piramit Ağı (SA-ROPA) gibi çeşitli

yöntemler kullanılmıştır. [15], Benjamin Staar, Michael Lütjena ve Michael Freitaga tarafından yürütülen "Endüstriyel yüzey denetimi için konvolüsyonel sinir ağları ile anomali tespiti" başlıklı Evrişimsel Sinir Ağı (ESA) [16], Oliver Ripple, Maximilian Muller ve Dorit Merhof tarafından "Kumaşlarda Anomali Tespiti için GAN Tabanlı Defekt Sentezi" başlıklı makaleyle yürütülen Generative Adversarial Network (GAN) [17], Domen Tabernik, Samo Šela, Jure Skvarc ve Danijel Skocaj tarafından "Yüzey kusurlarının tespiti için segmentasyon tabanlı derin öğrenme yaklaşımı" başlıklı makale ile yürütülen segmentasyon tabanlı derin öğrenme [18].

1.2. Tezin Amacı

Bu araştırmada, Autoencoders mimarisini kullanarak mermer yüzeylerindeki anomalileri tespit etmeyi amaçlayan bir algoritma oluşturduk. Bu platform, mermer üreten fabrikalar için mermer yüzeylerinin kontrol kalitesini artırmak amacıyla kullanılabilir. Mermerlerini test etmek isteyen bir şirket veya kişi varsa, test etmek istedikleri mermerin yüzeyinin fotoğrafını çekerler, daha sonra fotoğraf sisteme girilir, ardından sistem mermer yüzeyinde (çatlaklar, noktalar, eklemler vb.) gibi anormallikler veya kusurlar olup olmadığını işler, daha sonra sistem sonuçları görüntüler şeklinde verir ve mermeri iyi veya kötü kategorilere ayırmaktadır.

2. KAYNAK ARAŞTIRMASI

2.1. Mermer

Yukarıda açıklandığı gibi mermer, ısı, basınç ve sulu çözeltilerin (çoğunlukla kalsit (CaCO3) veya dolomit (CaMg(CO3)2)) etkisi altında yeniden kristalleşen karbonat minerallerinden oluşan metamorfik bir kayaçtır ve değişen kalınlıkta kristal bir dokuya sahiptir [19]. Bazı istisnalar olsa da mermer genellikle yapraksızdır (çok katmanlı).

Şekil 2.1. Mermer örneği

Mermer kelimesi Yunanca Marmaron kelimesinden türemiştir ve kökeni marmaros, parlayan taş, kristal taş, muhtemelen marmairo, parlamak veya ışıldamak fiilinden gelmektedir. bu aynı zamanda mermer benzeri anlamına gelen İngilizce marmoreal kelimesindeki mermer kelimesinin de temelidir. Jeolojide mermer terimi başkalaşmış kireçtaşını ifade eder, ancak taş işçiliğindeki kullanımı daha geniş olarak başkalaşmamış kireçtaşını içerir [20].

Mermerdeki ana mineral kalsittir ve bu mineralin sertlik, ışık geçirgenliği ve diğer özellikleri açısından çeşitlilik göstermesi, bazı mermerlerin hazırlanmasında birçok pratik sonuç doğurur. Kalsit kristalleri çift kırıcıdır - ışığı iki yönde ve bir yönde daha fazla iletirler; yarı saydamlığın önemli olduğu yerlerde kullanılmak üzere hazırlanan plakalar bu yöne paralel olarak kesilir. Mermer plakaların bükülmesi, ısıtma sırasında kalsit kristallerinin yönlü termal genleşmesine bağlanmıştır.

Atık mermer çamuru su bazlı boyalarda mineral dolgu maddesi olarak kullanılabilir [21]. Öğütülmüş kalsiyum karbonatın boya üretiminde dolgu maddesi olarak kullanılması boyanın parlaklığını, örtücülüğünü ve uygulama performansını artırabilir ve ayrıca titanyum dioksit gibi pahalı pigmentlerin yerini alabilir [21]. Mermer atıklarının geri dönüştürülmesi, büyük miktarda atığın depolanmamasını sağlayarak çevre kirliliğini azaltmakta ve böylece mermerin sürdürülebilirliğini gerçekleştirmektedir. Ekonomik gelir elde etmek için atıkların dönüştürülmesi ve zarar görmüş arazilerin restore edilmesi çevreyi iyileştirebilir.

Aşağıdaki tablo mermerin özelilliklerinin özetidir [22].

Tablo 2.1. Mermerin özellikleri

Renk	**Doku**	**Büyük Boy (Grand Size)**	**Mineraloji**	**Sertlik (Hardness)**	**Diğer Özellik**
Beyaz, Pembe, Siyah vb..	Granül (Granular)	Orta taneli (grained)	Kalsit	sert	Genellikle dokunmak cesur (gritty to touch)

2.1.1. Mermer kullanımı

Başlıca mermer türleri doğal mermer ve sentetik mermerdir. Doğal mermer, doğada kalsitten oluşan metamorfik taş formunda bulunur. Doğal mermer, yapı malzemeleri, heykeller, dış duvarlar, tezgahlar, merdivenler ve diğerlerinde yaygın olarak kullanılmaktadır. Sentetik mermer veya yapay mermer, küçük mermer molozu, taş tozu, plastik, çimento ve kum karışımından yapılır. Yapay mermer çeşitli renk ve desenlerde mevcuttur ve evlerin dış cephelerinde, mutfaklarda, duvarlarda, zeminlerde ve diğerlerinde bir yapı malzemesi olarak yaygın şekilde kullanılmaktadır [23].

Mermer üretiminin çoğunluğu bina ve inşaat için kullanılmaktadır. Kırılmış mermer yollar, beton ve demiryolu rayları için kullanılır. Boyut taşı, mermer gibi, taşın levhalar veya bloklar halinde kesilmesiyle yapılır. Boyut taşı inşaat, heykel, kaldırım ve anıtlarda kullanılır. Ön cephe Colorado mermerinden, zemin pembe Tennessee mermerinden inşa edilmiş ve Lincoln heykeli beyaz Georgia mermerinden yontulmuştur. Aşınma ve asit yağmurları mermer için çok zararlıdır.

2.2. Anomali Tespiti

Anomali tespiti, verilerde beklenen davranışa uymayan örüntüleri bulma problemini ifade eder. Bu uyumsuz örüntüler genellikle farklı uygulama alanlarında anomaliler, aykırı değerler, diskodantlar, gözlemler, istisnalar, sapmalar, sürprizler, tuhaflıklar veya kirleticiler olarak adlandırılır. Bunlardan anomali ve aykırı değer, anomali tespiti bağlamında en sık kullanılan iki terimdir; bazen birbirinin yerine kullanılır. Anomali tespiti, kredi kartları, sigorta veya sağlık hizmetleri için dolandırıcılık tespiti, siber güvenlik için izinsiz giriş tespiti, güvenlik açısından kritik sistemlerde hata tespiti ve düşman faaliyetleri için askeri gözetim gibi çok çeşitli uygulamalarda yaygın olarak kullanılmaktadır.

Verilerdeki anormalliklerin veya aykırı değerlerin tespit edilmesi 19. yüzyıldan beri istatistik camiasında çalışılmaktadır [24]. Zaman içinde, belirli uygulama alanları için özel olarak çeşitli anomali tespit teknikleri geliştirilirken, diğerleri daha genelleştirilmiştir.

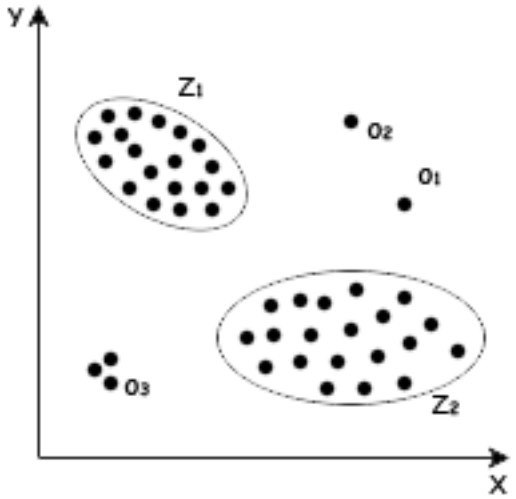
Anomali tespitinin önemi, verilerdeki anomalilerin çeşitli uygulama alanlarında önemli (ve genellikle kritik) eyleme geçirilebilir bilgilere dönüşmesinden kaynaklanmaktadır. Örneğin, uzay aracı sensörlerinden gelen anormal okumalar bazı uzay aracı bileşenlerinde hasar olduğuna işaret edebilir [25]. Anormal bir MRI görüntüsü kötü huylu bir tümörün varlığına işaret edebilir [26]. Bir bilgisayar ağındaki anormal trafik modelleri, saldırıya uğramış bir bilgisayarın hassas verileri yetkisiz bir hedefe gönderdiği anlamına gelebilir [27] veya kredi kartı işlem verilerindeki anormallikler kredi kartı veya kimlik hırsızlığına işaret edebilir [28].

2.2.1. Anomali tanıma

Veri Biliminde Anomaliler, veri setindeki diğer öğelerin beklenen modeline uymayan veri noktaları (genellikle çoklu noktalar olarak adlandırılır) olarak adlandırılır. Anomaliler, bir dağılım içinde meydana gelen farklı dağılımlar olarak adlandırılır. Verilerdeki anormallikler, çeşitli uygulama alanlarında harekete geçirilebilecek önemli (ve genellikle kritik) bilgilere dönüşür.

İstatistikte anomali ya da genel olarak aykırı değer olarak adlandırılan şey, diğer verilerin çoğundan çok farklı bir ağırlığa sahip olan veri ya da veri kümesidir. Başka bir deyişle, anormal veriler bir veri dağılımının ortalaması ve medyanından çok büyük

bir uzaklığa sahiptir. İki boyutlu bir düzlemde gösterilirse, anomali aşağıdaki gibi görselleştirilebilir.



Şekil 2.2. İki boyutlu bir veri setinde basit bir anormallik örneği.

Yukarıdaki Şekil 2.2'de O3, O2 ve O1 noktaları anormal verilerdir çünkü bu üç veri noktası Z1 ve Z2 normal veri gruplarından çok uzakta yer almaktadır.

2.2.2. Anomali tespitinin tarihçesi

Anomali tespitinin kritik bir bileşeni olan izinsiz giriş tespiti kavramı zaman içinde önemli ölçüde gelişmiştir. Başlangıçta, sistem yöneticilerinin tatil yapan kullanıcı hesaplarına erişilmesi veya beklenmedik yazıcı etkinliği gibi olağandışı etkinlikleri izledikleri manuel bir süreçti. Bu yaklaşım ölçeklenebilir değildi ve kısa süre sonra yerini kötü niyetli davranış belirtileri için denetim günlüklerinin ve sistem günlüklerinin analizine bıraktı [29].

1970'lerin sonu ve 1980'lerin başında, bu günlük analizleri, veri hacmi gerçek zamanlı izleme için pratik olmadığından, olayları araştırmak için çoğunlukla geriye dönük olarak kullanıldı. Dijital depolamanın uygun maliyetli olması, denetim günlüklerinin çevrimiçi olarak analiz edilmesine ve verileri incelemek için özel programlar geliştirilmesine yol açtı. Ancak bu programlar hesaplama yoğunluğu nedeniyle genellikle mesai saatleri dışında çalıştırılmıştır [29].

1990'lar, denetim verilerini üretildikleri anda analiz edebilen gerçek zamanlı izinsiz giriş tespit sistemlerinin ortaya çıkmasını sağlayarak saldırıları anında tespit etmeyi ve bunlara yanıt vermeyi mümkün kıldı. Bu, proaktif izinsiz giriş tespitine doğru önemli bir kaymaya işaret ediyordu [29].

Alan geliştikçe odak noktası, büyük ve karmaşık ağ ortamlarında verimli bir şekilde uygulanabilen, sürekli gelişen güvenlik tehditleri yelpazesine ve modern bilgi işlem altyapılarının dinamik doğasına uyum sağlayan çözümler yaratmaya kaymıştır.

2.2.3. Anomali tespit teknikleri

Anomali tespit teknikleri, etiketlerin kullanılabilirliğine bağlı olarak genel olarak üç ana kategoride sınıflandırılabilir: denetimsiz, yarı denetimli ve denetimli anomali tespiti [30] [31]. Sınıf etiketleri normal ya da anormal olabilir.

2.2.3.1. Denetimsiz anomali tespiti (unsupervised anomaly detection)

Bu yöntemler herhangi bir eğitim etiketi gerektirmez, bu da onları en esnek hale getirir. Verilerin anormal örneklerden çok daha fazla normal örnekle ağır bir şekilde çarpık olduğu varsayımıyla verilerin sadece içsel özelliklerini kullanırlar. [32]'da yazarlar, k-ortalamalar kümeleme ve grafik tabanlı algoritma kullanarak meyve kusurlarının tespiti için bir görüntü bölütleme yaklaşımı araştırmışlardır. Yöntem, bölütleme işlemi için yavaş bir bölge büyütme tekniği kullanmıştır. Ayrıca, ilk küme merkezlerinin seçimine büyük ölçüde bağımlıydı ve farklı seçimlere dayalı olarak farklı sonuçlar elde etti. Denetimsiz yaklaşımlarla, belirli anormallik türlerinin hedeflenmesi umulamaz çünkü etiketleme yoktur. Ayrıca, denetimsiz teknikler etiketleme gereksinimleri açısından en fazla esnekliği sunsa da, karmaşık ve yüksek boyutlu uzaylarda ortak noktaları öğrenmekte genellikle zorlanırlar [30].

2.2.3.2. Yarı denetimli anomali tespiti (semi-supervised anomaly detection)

Bu teknikler için temel varsayım, yalnızca normal sınıf için etiketlenmiş eğitim örneklerine erişimimiz olduğudur. Bu nedenle, bu teknikler denetimsiz tekniklere göre daha az esnektir. Bunlar, verilerin altında yatan normal sınıf dağılımını tahmin etmeye veya modellemeye çalışır. Bunu, örneklerin öğrenilen dağılımdan sapmasının bir tür ölçümü ve bir eşiğe dayalı sınıflandırma takip eder. Schlegl ve diğerleri. [33] retinanın optik koherens tomografi görüntülerinde anomali tespiti için Generative Adversarial Networks (GANs) kullanmıştır. Anatomik değişkenliği kodlamak üzere altta yatan dağılımı öğrenmek için normal veriler üzerinde bir GAN eğitmişlerdir. Ancak, giriş görüntüsünü gizli uzaya eşlemek için bir kodlayıcı eğitmemişlerdir. Sonuç olarak, yöntemin her sorgu görüntüsü için gizli uzayda görsel olarak en benzer oluşturulan görüntüye karşılık gelen bir nokta bulmak için bir optimizasyon adımına ihtiyacı vardı. Bu da tekniğin hesaplama açısından pahalı olmasına neden olmuştur. Model, sorgu

görüntüsünün optimizasyondan elde edilen GAN tarafından üretilen görüntüden çıkarılmasıyla elde edilen artık görüntü olarak bir anormallik puanı ve bir segmentasyon çıktısı vermiştir. Zenati ve diğerleri [34] veri uzayından gizli uzaya eşlemeyi öğrenen bir kodlayıcıyı eğiterek hız darboğazının üstesinden gelmeye çalışmıştır. Bunu sadece iki görüntü veri kümesi SVHN ve CIFAR10 üzerinde test ettiler ve sonuçlar iyi görünmüyordu. Sırasıyla 0.5753 ve 0.6072 AUROC değerleri elde etmişlerdir ki bu değerler rastgele sınıflandırma modeli değeri olan 0.5'in marjinal olarak üzerindedir. [35]'de yazarlar, sıcak haddelenmiş çelik şerit yüzeylerinin kusur segmentasyonu için konvolüsyonel otomatik kodlayıcılar kullanmışlardır. Sadece normal görüntüler üzerinde bir otomatik kodlayıcıyı eğitmişlerdir. Anormallik tespit süreci, artık görüntünün (yeniden yapılandırılmış görüntünün giriş görüntüsünden çıkarılmasıyla elde edilen) yeniden yapılandırılmış görüntüye ağırlıklı olarak eklenmesi olan bir keskinleştirme adımını içeriyordu. Bunu Gauss bulanıklaştırma ve eşikleme takip etmiştir. Hiçbir nicel sonuç rapor edilmemiştir. Ancak, nitel sonuçlar tekniğin son derece gürültülü olduğunu göstermiştir. Aydınlatma değişikliklerini bile istenmeyen ve sonuçta performansı etkileyen kusurlu bölgeler olarak bölütlemiştir.

2.2.3.3. Denetimli anomali tespiti (supervised anomaly detection)

Denetimli anomali tespiti. Denetimli modda eğitilen teknikler, normal ve anomali sınıfları için örneklerle etiketlenmiş eğitim veri kümelerinin mevcut olduğunu varsayar. Bu tür durumlarda yaygın bir yaklaşım, normal ve anomali sınıfı için bir tahmin modeli oluşturmaktır. Görülmeyen her veri örneği, hangi sınıfa ait olduğunu belirlemek için modelle karşılaştırılır. Denetimli anomali tespitinde ortaya çıkan iki ana sorun vardır. İlk olarak, eğitim verisinde normal örneklere kıyasla çok daha az sayıda anomali örneği bulunmaktadır. Dengesiz sınıf dağılımı nedeniyle ortaya çıkan sorunlar veri madenciliği ve makine öğrenimi literatüründe tartışılmıştır [36, 37, 38, 39]. İkinci olarak, özellikle anomali sınıfları için doğru ve temsili etiketler elde etmek genellikle oldukça zordur. Etiketli bir eğitim veri seti elde etmek için normal bir veri setine yapay anomaliler enjekte eden bir dizi teknik önerilmiştir [40] [41] [42].

2.2.4. Anormallik tespiti uygulamaları

Anormallik tespiti çok sayıda ve çeşitli alanlarda uygulanabilir ve denetimsiz makine öğreniminin önemli bir alt alanıdır. Bu nedenle anormallik tespitinin siber güvenlik, izinsiz giriş tespiti, sahtekarlık tespiti, hata tespiti, sistem sağlığı izleme, sensör ağlarında olay tespiti, ekosistem bozukluklarının tespiti, makine görüşü kullanılarak

görüntülerdeki kusurların tespiti, tıbbi teşhis ve kanun yaptırımı alanlarında uygulamaları vardır.

İzinsiz giriş tespit sistemleri (IDS) için anormallik tespiti, 1986 yılında Dorothy Denning tarafından önerildi [43]. IDS için anormallik tespiti genellikle eşikleme ve istatistiklerle yapılır, ancak aynı zamanda yazılımsal hesaplama ve tümevarımsal öğrenmeyle de yapılabilir [44]. 1999'da önerilen özellik türleri arasında kullanıcı profilleri, iş istasyonları, ağlar, uzak ana bilgisayarlar, kullanıcı grupları ve frekans, ortalama, varyans, kovaryans ve standart sapmaya dayalı programlar yer alıyordu. İzinsiz giriş tespitinde anormallik tespitinin karşılığı kötüye kullanım tespitidir.

Veri analizi alanında, anormallikleri ortadan kaldırmak için verilerin ön işlenmesi önemli bir adım olabilir ve bu, çeşitli nedenlerden dolayı yapılır. Anormallikler ortadan kaldırıldığında ortalama ve standart sapma gibi istatistikler daha doğru hale gelir ve veri görselleştirmesi de geliştirilebilir. Denetimli öğrenmede, anormal verilerin bir veri kümesinden kaldırılması çoğu zaman doğrulukta istatistiksel olarak önemli iyileşmeler sağlar [45].

Daha sonra, video gözetimi alanında anormallik tespiti, güvenliği ve güvenliği artırmak için giderek daha önemli hale geliyor [46, 47]. Derin öğrenme teknolojisinin ilerlemesiyle birlikte, Evrişimli Sinir Ağları (ESA'lar) ve Basit Tekrarlayan Birimler'i (BTB'ler) kullanan yöntemler, video verilerindeki olağandışı aktivite veya davranışların belirlenmesinde önemli umut vaat ediyor. Model, potansiyel güvenlik tehditlerine veya güvenlik ihlallerine işaret edebilecek normdan sapan kalıpları tanıyarak kapsamlı video akışlarını gerçek zamanlı olarak işleyebilir ve analiz edebilir [46].

Nesnelerin İnterneti (IoT) alanında anormallik tespiti sistem güvenliği ve verimliliği açısından çok önemlidir. IoT cihazlarından oluşan karmaşık ağlardaki sistem arızalarının ve güvenlik ihlallerinin tespit edilmesine yardımcı olur [48]. Yöntemin gerçek zamanlı verileri, birden fazla cihaz türünü yönetmesi ve etkili bir şekilde ölçeklendirmesi gerekir. Garbe ve arkadaşlar [49] mekansal kümelemeyi, yoğunluğa dayalı kümelemeyi ve yerelliğe duyarlı karma oluşturmayı birleştirerek geleneksel yöntemleri geliştiren çok aşamalı bir anormallik tespit çerçevesi sunmuştur. Bu özel yaklaşım, IoT verilerinin geniş ve çeşitli yapısını daha iyi ele almak ve böylece akıllı

altyapı ve endüstriyel IoT sistemlerinde operasyonel güvenliği ve güvenilirliği artırmak için tasarlanmıştır.

3. KULLANILAN TEKNOLOJİ VE YÖNTEMLER

3.1. Kullanılan Yazılımlar

Bu araştırmada yapılan algoritmayı gerçekleştirmek için programlama dili olarak python kullanılmıştır. Python, kullanıcıların bu programlama dilini kullanmasını kolaylaştırabilecek çeşitli uygulamalar için binlerce kütüphaneye sahiptir. Ayrıca, Spyder da bir Entegre Geliştirme Ortamı (EGO) olarak kullanılmaktadır.

3.1.1. Python

Python yorumlanmış, etkileşimli ve nesne yönelimli bir programlama dilidir. Modüller, istisnalar, dinamik yazım, yüksek seviyeli dinamik veri türleri ve sınıflar içerir. Nesne yönelimli programlamanın ötesinde prosedürel ve fonksiyonel programlama gibi çeşitli programlama paradigmalarını destekler. Python muazzam gücü çok açık bir sözdizimiyle birleştirir. Birçok sistem çağrısı ve kütüphanenin yanı sıra çeşitli pencere sistemleri için arayüzlere sahiptir ve C veya C++ ile genişletilebilir. Programlanabilir arayüzler gerektiren uygulamalar için bir uzantı dili olarak da kullanılabilir [50].

Çok yönlülüğü ve kullanım kolaylığı nedeniyle en yaygın kullanılan programlama dili haline gelmiştir. Özellikle yeni başlayanlar için. Python son yıllarda dünyanın çeşitli yerlerinde kullanılan en popüler programlama dili haline gelmiştir. Makine öğreniminden, yazılım testlerine ve web sitelerine kadar çeşitli ihtiyaçlar için kullanılabilir.

Stack Overflow'un 2022 geliştirici anketine göre [51], Python en popüler dördüncü programlama dilidir. Ankete katılanların neredeyse %50'si çalışma zamanlarının neredeyse yarısını bu programlama dilini kullanarak geçirdiklerini belirtmiştir.

3.1.1.1. Python'un tarihçesi

Python programlama dili Hollandalı bir bilgisayar programcısı olan Guido Van Rossum [52] tarafından yaratılmıştır. 1989 yılında Centrum Wiskunde & Informatica'da (CWI) Noel zamanı meşgul olmak için bir hobi projesi olarak başladı. Python ismi Monty Python'dan gelmektedir. Guido van Rossum bunu yarattığında BBC'nin Monty Python's Flying Circus senaryosunu da okuyordu. Bu ismin kısa ve

biraz da gizemli olduğunu düşünmüş. Bu nedenle yaratıcı, yarattığı programlama dili için bu ismi kullanmayı seçtimiştir. İşte python sürümlerinin tarihçesi:

- Guido Van Rossum Python kodunun ilk sürümünü (0.9.0 sürümü) 1991 yılında yayınladı. Bu sürüm, bazı veri türleri ve hataları ele alan fonksiyonlar gibi iyi özelliklere sahipti.
- Python 1.0, veri giriş sürecini kolaylaştırmak için eşleme, filtreleme ve azaltma gibi yeni işlevlerle 1994 yılında yayınlandı.
- Python 2.0, Unicode karakter desteği ve daha kısa döngü listeleri gibi programcılar için yararlı yeni özelliklerle 16 Ekim 2000'de piyasaya sürüldü.
- 3 Aralık 2008'de Python 3.0 piyasaya sürüldü. Bu sürüm, bir yazdırma işlevi ve sayı bölme ve sorun işleme için daha fazla destek gibi özellikler içeriyordu.

3.1.1.2. Python kütüphanesi

Kütüphaneler, geliştiricilerin sıfırdan kod yazmaktan kaçınmak için Phyton programlarına dahil edebilecekleri sık kullanılan kod koleksiyonlarıdır. Phyton varsayılan olarak, birçok yeniden kullanılabilir işlev içeren Standart Kütüphane ile birlikte gelir. Ayrıca, web geliştirme, veri bilimi ve makine öğrenimi dahil olmak üzere çeşitli uygulamalar için 137.000'den fazla Phyton kütüphanesi mevcuttur. Veri işlemede kullanılan bazı yaygın kütüphaneler şunlardır:

- Matplotlib

  Geliştiriciler, verileri yüksek kaliteli iki ve üç boyutlu (2D ve 3D) grafiklerde çizmek için Matplotlib'i kullanır. Matplotlib genellikle bilimsel uygulamalarda kullanılır. Matplotlib ile verileri çubuk grafikler ve çizgi grafikler gibi farklı grafiklerde görüntüleyerek görselleştirebilirsiniz. Ayrıca aynı anda birden fazla grafik çizebilirsiniz ve grafikler platformlar arasında taşınabilir.
- Pandas

  Pandas, zaman serisi verilerini ve tablolar ve seriler gibi yapılandırılmış verileri işlemek için kullanabileceğiniz optimize edilmiş esnek veri yapıları sağlar. Örneğin, Pandas'ı verileri okumak, yazmak, birleştirmek, filtrelemek ve gruplamak için kullanabilirsiniz. Birçok kişi Pandas'ı veri bilimi, veri analizi ve makine öğrenimi (ML) görevleri için kullanmaktadır.
- NumPy

  NumPy, geliştiricilerin dizileri kolayca oluşturmak ve yönetmek, mantıksal

formları manipüle etmek ve doğrusal cebir işlemleri gerçekleştirmek için kullandıkları popüler bir kütüphanedir. NumPy, C ve C++ gibi birçok dil ile entegrasyonu destekler.

- OpenCV-Python

  OpenCV-Python, geliştiricilerin bilgisayarla görme uygulamaları için görüntüleri işlemek üzere kullandıkları bir kütüphanedir. OpenCV-Python, görüntüleri aynı anda okuma ve yazma, 2D'den 3D ortamlar oluşturma ve videodan görüntüleri yakalama ve analiz etme gibi görüntü işleme görevleri için birçok işlev sağlar.

- Keras

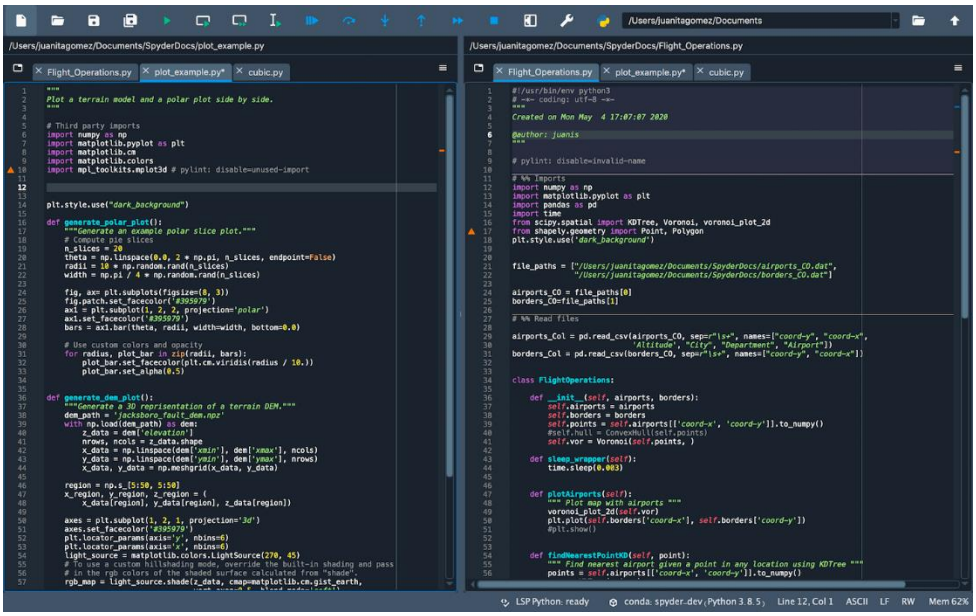
  Keras, Python'un veri işleme, görselleştirme ve daha fazlası için mükemmel desteğe sahip derin sinir ağı kütüphanesidir. Keras birçok sinir ağını destekler. Keras, yenilikçi uygulamalar yazmada esneklik sunan modüler bir yapıya sahiptir.

3.1.2. Sypder

Spyder, Python için Python'da yazılmış ve bilim insanları, mühendisler ve veri analistleri tarafından ve onlar için tasarlanmış güçlü bir bilimsel ortamdır. Kapsamlı bir geliştirme aracının gelişmiş düzenleme, analiz, hata ayıklama ve profil oluşturma işlevselliği ile bilimsel bir paketin veri keşfi, etkileşimli yürütme, derin araştırma ve güzel görselleştirme yeteneklerinin benzersiz bir kombinasyonunu temsil eder. Spyder IDE'ye aşağıdaki bileşenler dahildir [53]:

- Editor

  Spyder'ın çok dilli Editör paneli, kaynak dosyaları oluşturmak, açmak ve değiştirmek için bir yer olan IDE'nin önemli bir unsurudur. Editör, otomatik tamamlama, gerçek zamanlı analiz, sözdizimi vurgulama, yatay ve dikey bölme ve daha fazlası gibi çok çeşitli temel özellikler sunar. Buna ek olarak, kullanımı kolay ve verimli bir düzenleme deneyimi için bir dizi gelişmiş aracı entegre eder.

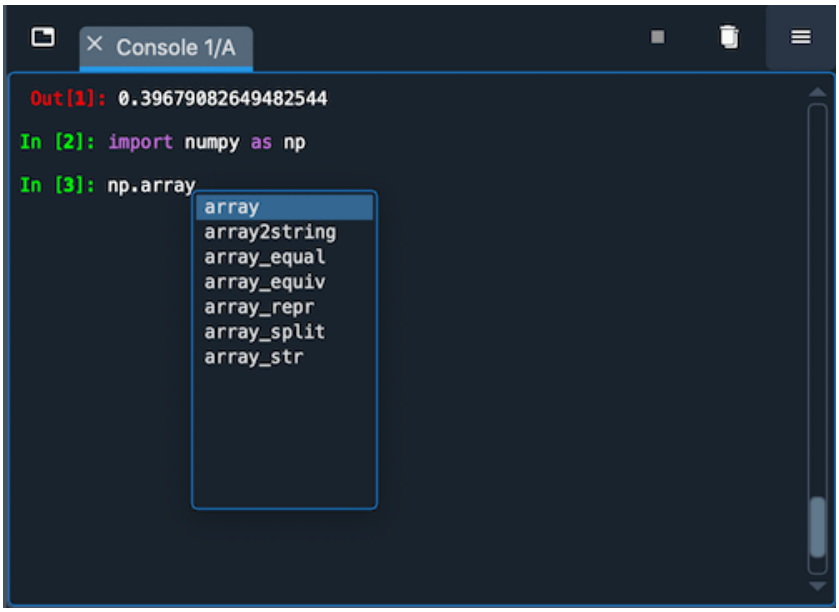


Şekil 3.1. Sypder'de editör paneli örneği

- IPython Konsolu (IPython Console)

IPython konsolu, komutları çalıştırmamıza ve IPython yorumlayıcısı içindeki verilerle etkileşime girmemize olanak tanır. Tek bir GUI'de istediğiniz kadar IPython konsolunun gücünden yararlanın. Satır, hücre veya dosya bazında kod çalıştırın veya hata ayıklama, grafikler ve sihirli yorumlarla etkileşimli olarak çalışın.

Ayrıca Spyder da birkaç özel konsol türünü de destekler. Sympy konsolu, Spyder içinde sembolik matematik ifadeleri oluşturmanıza ve görüntülemenize olanak tanır. Cython konsolu, kodunuzu hızlandırmak için Cython dilini kullanmanıza ve C işlevlerini doğrudan Python'dan çağırmanıza olanak tanır. Son olarak, Pylab konsolu varsayılan olarak yaygın Numpy ve Matplotlib işlevlerini yükler; bunlar kullanımdan kaldırılmış ve yeni kodlar için kesinlikle önerilmese de, bunları gerektiren eski komut dosyaları için gerekirse hala kullanılabilirler.

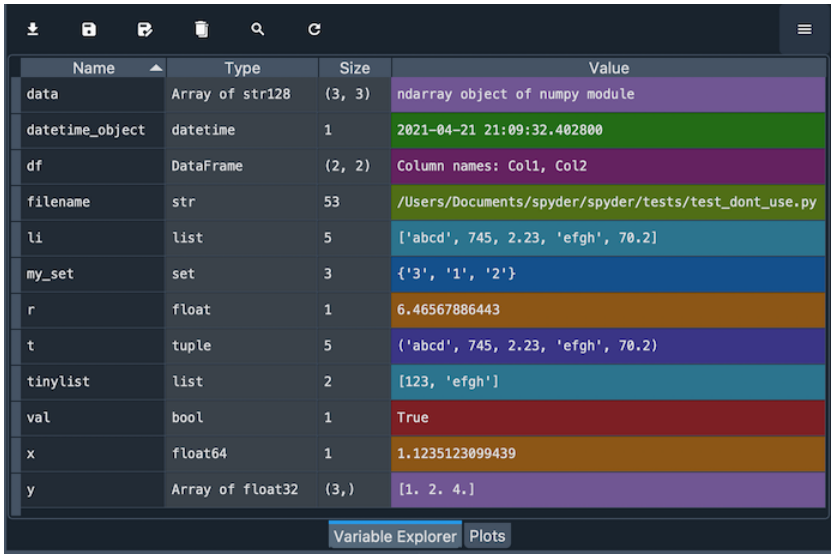


Şekil 3.2. Sypder üzerinde IPython Konsol örneği

- Değişken Gezgini (Variable Explore)

Değişken Gezgini, kod çalıştırırken oluşturulan nesnelere etkileşimli olarak göz atmanıza ve bunları yönetmenize olanak tanır. O anda seçili olan IPython Console oturumunun ad alanı içeriğini (tüm global nesneler, değişkenler, sınıf örnekleri ve daha fazlası dahil) gösterir ve çeşitli GUI tabanlı düzenleyiciler aracılığıyla değerlerini eklemenize, silmenize ve düzenlemenize olanak tanır.

Değişken Gezgini size her nesnenin adı, boyutu, türü ve değeri hakkında bilgi verir. Sayı, dize veya boolean gibi bir skaler değişkeni değiştirmek için panele çift tıklamanız ve yeni değerini yazmanız yeterlidir.

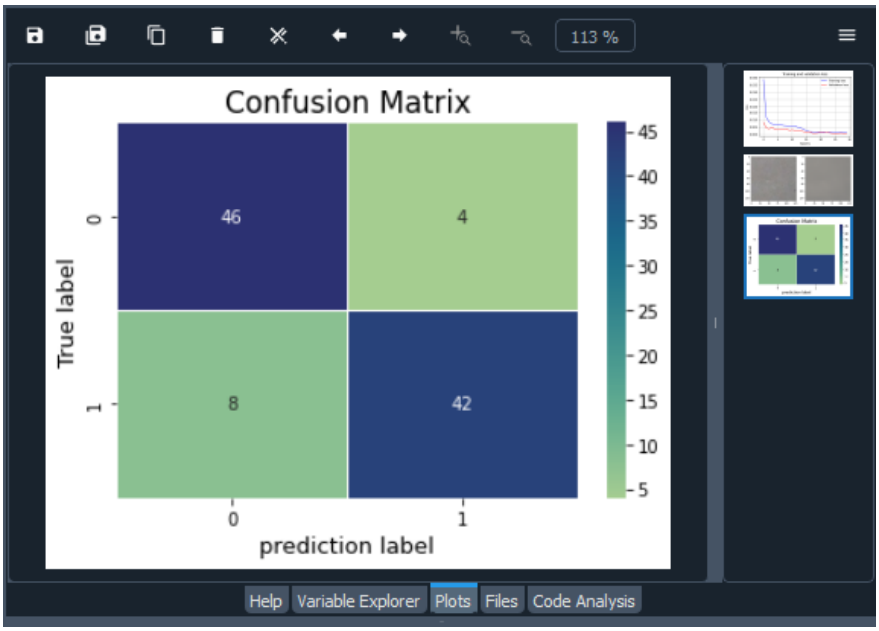


Şekil 3.3. Sypder'da değişken keşfetme örneği

- Arsalar (Plots)

Çizimler paneli statik görüntüleri ve oturumumuz sırasında oluşturulan görüntüleri gösterir. Bu panel, IPython Konsolundan, Editördeki kodumuz tarafından oluşturulan veya Değişken Gezgini tarafından oluşturulan ve onlarla bir şekilde etkileşime girmemizi sağlayan grafikleri görüntüleyecektir.

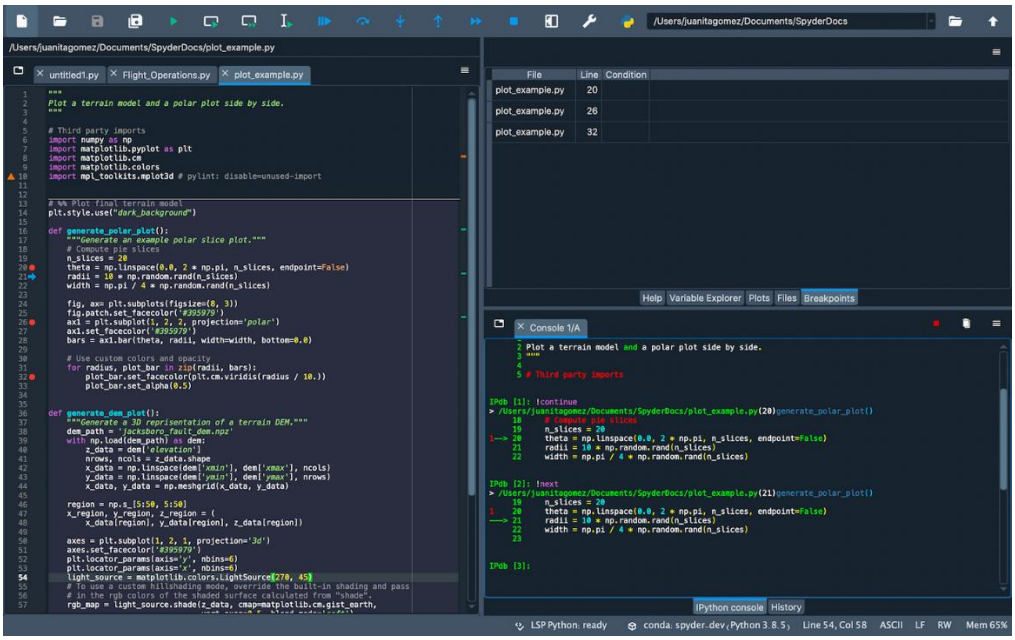
Çizimler panelinde görüntülenen görüntüler, o anda etkin olan Konsol sekmesiyle ilişkili olanlardır; konsol değişirse, görüntülenen çizimlerin listesi değişecektir (veya yeni bir konsol ise hiç olmayacaktır).



Şekil 3.4. Spyder üzerinde Plot arayüzü örneği

- Hata Ayıklayıcı (Debugger)

Spyder'da hata ayıklama, IPython Konsolundaki geliştirilmiş ipdb hata ayıklayıcısıyla entegrasyon yoluyla desteklenir. Bu, kesme noktalarının ve yürütme akışının doğrudan Spyder GUI'den ve tüm tanıdık IPython konsol komutlarıyla görüntülenmesine ve kontrol edilmesine olanak tanır. Hata ayıklayıcı yürütmesini, standart IPython konsol komutlarının yanı sıra Hata Ayıklama menüsünden, Hata Ayıklama araç çubuğundan ve yapılandırılabilir klavye kısayollarından tamamen kontrol edebiliriz. Spyder'ın hata ayıklayıcısı, normal bir etkileşimli yorumlayıcıda olduğu gibi çalışan sözdizimi vurgulama, kod tamamlama ve komut geçmişi sunar.



Şekil 3.5. Sypder'da hata ayıklama örneği

- Yardım panelini (Help)

Yardım panelini, modüller, sınıflar, işlevler ve yöntemler dahil olmak üzere docstring içeren herhangi bir nesne için zengin belgeleri bulmak, işlemek ve görüntülemek için kullanabiliriz. Bu, iş akışınızı kesintiye uğratmak zorunda kalmadan belgelere doğrudan Spyder'dan kolayca erişmenizi sağlar.

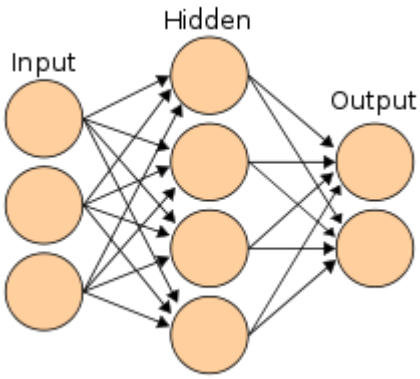
3.2. Yöntem

Bu araştırmada, yapay sinir ağlarını (YSA) içeren bir hesaplama süreci yoluyla veri karmaşıklığını azaltmak için kullanılabilen makine öğrenimi teknolojilerinden biri olan Otomatik Kodlayıcı mimarisi kullanılmıştır, Yapay sinir ağları, birbirine bağlı ve belirli sorunları çözmek için birlikte çalışan birçok nörondan oluşan derin öğrenmenin temel kavramıdır. Otomatik Kodlayıcı hakkında daha fazla bilgi edinmek için öncelikle yapay sinir ağının (YSA) ne olduğunu tartışmak faydalı olacaktır.

3.2.1. Yapay sinir ağlarını

Yapay sinir ağının (YSA) (ingilizcede: *artificial neural network;* ANN, veya *simulated neural network* (SNN), veya *neural network* (NN)), insan sinir sistemi üzerine modellenmiş küçük işlem birimlerinden oluşan bir ağdır. Bir YSA, ağ üzerinden akan harici ve dahili bilgilere dayanarak sorunları çözmek için yapısını

değiştirebilen (*adaptable*) uyarlanabilir bir sistemdir. Uyarlanabilir yapıları nedeniyle, YSA'lar genellikle uyarlanabilir ağlar olarak da adlandırılır. [54].



Şekil 3.6. Yapay sinir ağı örneği

Yapay Sinir Ağlarını, biyolojik bir beyindeki nöronları gevşek bir şekilde modelleyen yapay nöronlar adı verilen bağlı birimler veya düğümler koleksiyonuna dayanmaktadır. Her bağlantı, biyolojik beyindeki sinapslar gibi, diğer nöronlara sinyal iletebilir. Yapay nöron sinyali alır, sonra işler ve bağlı olduğu nöronlara sinyal gönderebilir. Bir bağlantıya giden "sinyal" gerçek bir sayıdır ve her nöronun çıktısı, girdilerinin toplamının doğrusal olmayan bir fonksiyonu ile hesaplanır. Bağlantılar kenar olarak adlandırılır. Nöronlar ve kenarlar genellikle öğrenme ilerledikçe ayarlanan ağırlıklara sahiptir. Bu ağırlıklar bir bağlantının sinyal gücünü artırır veya azaltır. Nöronların eşikleri olabilir, böylece sinyaller yalnızca toplam sinyal eşiği geçerse gönderilir.

Genellikle nöronlar birden fazla katman halinde birleştirilir. Farklı katmanlar girdileri üzerinde farklı dönüşümler gerçekleştirebilir. Sinyal, muhtemelen katmanlardan birkaç kez geçtikten sonra ilk katmandan (giriş katmanı) son katmana (çıkış katmanı) gider. Bir ağ genellikle en az 2 gizli katmana sahipse derin sinir ağı olarak adlandırılır.

3.2.1.1. Yapay sinir ağı modeli

Yapay sinir ağlarındaki model temel olarak $f : X \to Y$ fonksiyonunu tanımlayan matematiksel bir model fonksiyonudur. Yapay Sinir Ağlarındaki "Ağ" terimi, farklı katmanlara yerleştirilmiş birden fazla nöronun birbirine bağlanmasını ifade eder. Genel olarak, bir yapay sinir ağının katmanları üç bölüme ayrılır:

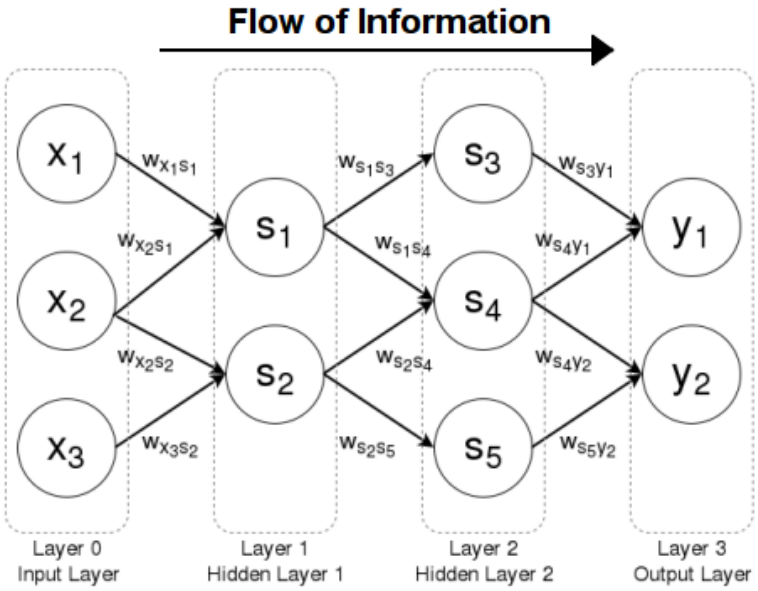
- Giriş katmanı (input layer), X değişkeninden giriş verisi alan nöronlardan oluşur. Bu katmandaki tüm nöronlar gizli katmandaki nöronlara veya ağ gizli bir katman kullanmıyorsa doğrudan çıktı katmanına bağlanabilir.

- Gizli katman (hidden layer), giriş katmanından veri alan nöronlardan oluşur.
- Çıkış katmanı (output Layer), gizli katmandan veya doğrudan giriş katmanından veri alan nöronlardan oluşur ve çıkış değeri X'ten Y değerine yapılan hesaplamanın sonucunu simgeler.

3.2.1.2. Yapay sinir ağı türleri

1) İleri Beslemeli (feedForward) Sinir Ağı:

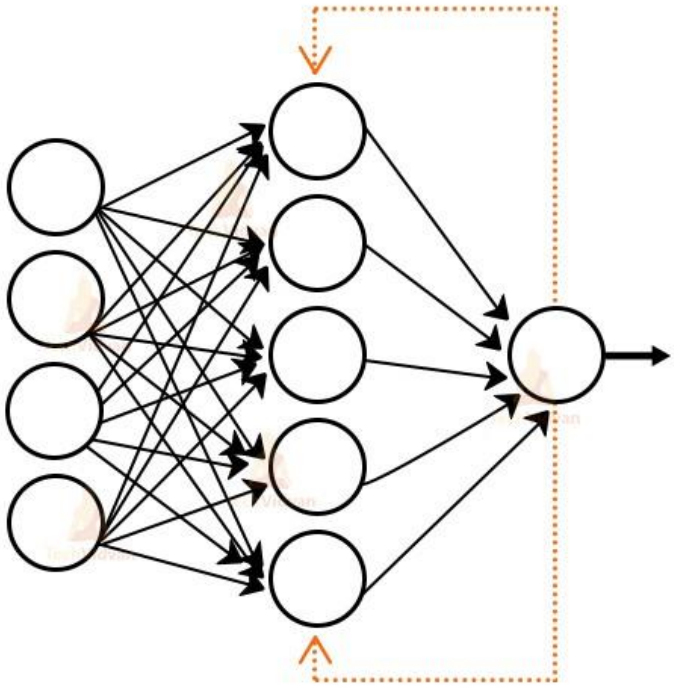
İleri beslemeli sinir ağlarında bilgi akışı tek yönlüdür. Yani veri giriş katmanından gizli katmana ve oradan da çıkış katmanına akar. Geri bildirim döngüsü yoktur. Bu sinir ağları, sınıflandırma ve görüntü tanıma gibi görevler için denetimli öğrenmede yaygın olarak kullanılır. Veriler sıralı olmadığında kullanırız. İleri beslemeli ağlar, evrişimli sinir ağları (CNN) ile karşılaştırılabilir.



Şekil 3.7. İleri Beslemeli (feedForward) Sinir Ağı

2) Geribildirim (feedBack) Sinir Ağı:

Geri besleme döngüsü, geri beslemeli YSA'nın bir öğesidir. Tekrarlayan sinir ağları gibi bu tür sinir ağları çoğunlukla hafızayı korumak için kullanılır. Bu ağ, verilerin sıralı veya zamana bağlı olduğu durumlarda en iyi şekilde kullanılır. Geri bildirim döngüleri tekrarlayan sinir ağlarını (RNN) tanımlar.



Şekil 3.8. Geribildirim (feedBack) Sinir Ağı

3.2.1.3. Yapay sinir ağı uygulaması

Yapay sinir ağı mimarisini uygulamak için kullanılan çeşitli alanlar aşağıda verilmiştir.

Konuşma tanıma alanı büyük ölçüde yapay sinir ağlarına (YSA) dayanmaktadır. Önceki konuşma tanıma modellerinde Gizli Markov Modelleri gibi istatistiksel modeller kullanılıyordu. Derin öğrenmenin kullanıma sunulmasıyla birlikte, bir tür sinir ağı, kesin sınıflandırma elde etmenin tek yolu haline geldi. Lim ve arkadaşları [55] yapay sinir ağlarının, özellikle Olasılıksal sinir ağı (OSA) modellerinin sesli konuşma sinyallerinin sınıflandırılmasında kullanımına ilişkin deneyler sunmuşlar ve tatmin edici sonuçlara ulaşmışlardır.

El Yazısı Karakter tanıma alanında, YSA el yazısı karakterleri tanımak için kullanılır. Elle yazılan karakterler harf veya sayı olabilir ve sinir ağları bunları tanıyacak şekilde eğitilmiştir. Vinita ve arkadaşlarının yaptığı gibi [56] ölçeklendirme dönüşümlerini, çeviriyi veya her ikisinin birleşimini gerçekleştirebilen bir el yazısı tanıma sistemi oluşturdular.

İmza Sınıflandırması yapay sinir ağları, bu kimlik doğrulama sistemi geliştirilirken imzaların tanınması ve kişinin sınıfına göre sınıflandırılması amacıyla kullanılmaktadır. Ayrıca sinir ağları bir imzanın gerçek olup olmadığını

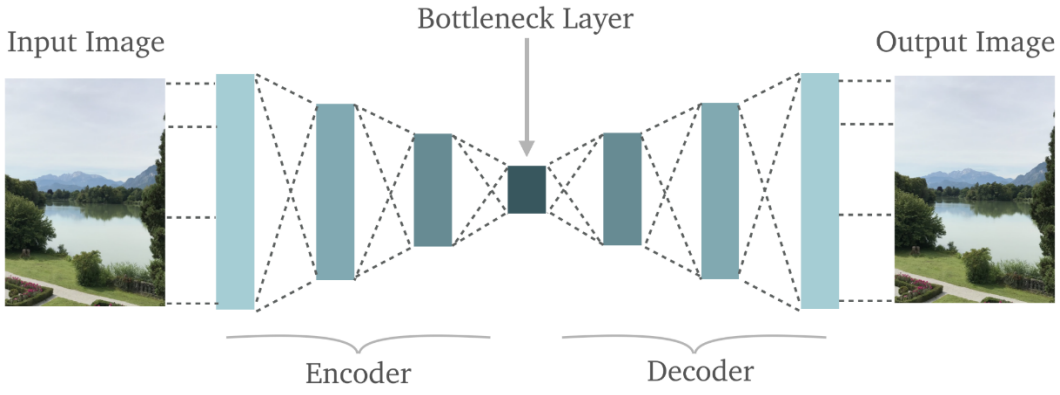
belirleyebiliyor. Ali ve arkadaşları imzaları [57] doğrulamak ve sınıflandırmak için yapay sinir ağlarını (YSA) kullandılar: doğru veya sahte ve %90 eşiğinin altında yaklaşık %93'lük bir sınıflandırma oranı elde ettiler.

Sağlık sektöründe kanser hücrelerini tespit etmek ve MRI görüntülerini analiz ederek detaylı sonuçlar sağlamak amacıyla yapay sinir ağları kullanılabilmektedir. Ganesan ve arkadaşları [58], çeşitli kanser türlerini teşhis etmek ve yalnızca hücre şekli bilgisini kullanarak yüksek derecede istilacı kanser hücre hatlarını daha az istilacı hatlardan ayırt etmek için yapay sinir ağlarını kullandılar.

Bunun dışında siber güvenlik alanında da güvenli ve tehlikeli faaliyetleri ayırt etmek amacıyla yapay sinir ağlarından faydalanılmaktadır. Örneğin, Android kötü amaçlı yazılımlarını sınıflandırmak [59], tehdit aktörlerine ait alanları belirlemek ve güvenlik riski oluşturan URL'leri tespit etmek için makine öğrenimi kullanılmıştır. Botnet'leri [60], kredi kartı sahtekarlığını ve ağ saldırılarını tespit etmek amacıyla sızma testi için tasarlanan YSA sistemleri üzerinde başka araştırmalar da yapılmıştır.

3.2.2. Otomatik kodlayıcı (autoencoder)

Otomatik kodlayıcı, etiketlenmemiş verilerin verimli bir şekilde kodlanmasını öğrenmek için kullanılan bir yapay sinir ağı türüdür (denetimsiz öğrenme) Otomatik kodlayıcılar iki işlevi öğrenir: Giriş verilerini dönüştüren bir kodlama işlevi ve giriş verilerini kodlanmış temsilden yeniden oluşturan bir kod çözme işlevi. Otomatik kodlayıcılar, genellikle boyutsallığı azaltmak amacıyla bir veri kümesi için verimli bir temsil (kodlama) öğrenir.



Şekil 3.9. Otomatik Kodlayıcı mimarisi örneği

Otomatik kodlayıcılar veri sıkıştırma, boyut azaltma veya anomali tespiti gibi çeşitli amaçlar için kullanılabilir. Bununla birlikte, diğer makine öğrenimi teknolojileri gibi, oto kodlayıcı da etkili bir şekilde çalışmak için yeterli eğitim verisi gerektirir.

3.2.2.1. Otomatik kodlayıcı mimarisi

Otomatik kodlayıcı mimarisinin 3 ana katmanı vardır [61]:

1. Kodlayıcı (Encoder): Eğit-doğrulama-test seti giriş verilerini, genellikle giriş verilerinden birkaç büyüklük sırası daha küçük olan kodlanmış bir temsile sıkıştıran bir modül.

2. Darboğaz (Boottleneck): Sıkıştırılmış bilgi temsilini içeren ve bu nedenle ağın en önemli parçası olan modül.

3. Kod Çözücü (Decoder): Ağın bilgi temsilini "açmasına" ve verileri kodlanmış halinden yeniden yapılandırmasına yardımcı olan modül. Çıktı daha sonra temel gerçekle karşılaştırılır.

Kodlayıcı katmanı giriş verilerini alır ve bunları daha küçük bir temsile dönüştürür veya yeniden yapılandırır (kodlama/*encoding*). Kod katmanı, kodlayıcı katmanından gelen sıkıştırılmış verileri depolar ve kod çözücü katmanı için giriş verisi olur. Kod çözücü katmanı, kod katmanından gelen verileri orijinaline benzer verilere yeniden yapılandırmaktan sorumludur (kod çözme/*decoding*).

3.2.2.2. Otomatik kodlayıcı türleri

Bu alanda birkaç yıl süren araştırmalar yapılmış ve oto kodlayıcıların birkaç farklı çeşidi sunulmuştur. Bu bölümde yaygın olarak kullanılan bazı oto kodlayıcı türlerini kısaca açıklayacağız.

- **Eksik otomatik kodlayıcı (undercomplete autoencoder)**

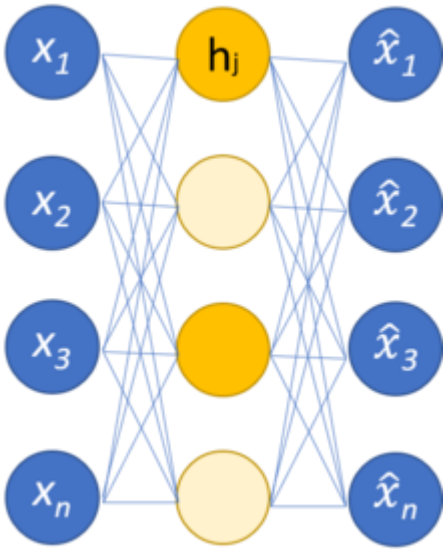
Eksik Otomatik Kodlayıcı, en basit otomatik kodlayıcı türlerinden biridir. Eksik Otomatik Kodlayıcı bir görüntüyü alır ve aynı görüntüyü çıktı olarak tahmin etmeye çalışır, böylece görüntüyü sıkıştırılmış darboğaz bölgesinden yeniden yapılandırır. Eksik Otomatik Kodlayıcı, herhangi bir etiket biçimini almadığı için tamamen denetimsizdir, hedef girdiyle aynıdır. Bu tür otomatik kodlayıcıların ana kullanımı, giriş verilerinin sıkıştırılmış bir suretini oluşturan ve gerektiğinde ağ yardımıyla kolayca geri açılabilen gizli alanların veya darboğazların oluşturulmasıdır.

Eksik bir Otomatik Kodlayıcıyı eğitmek için kullanılan kayıp fonksiyonuna yeniden yapılandırma kaybı (*reconstruction loss*) denir çünkü görüntünün giriş verilerinden ne kadar iyi yeniden yapılandırıldığını kontrol eder. Yeniden yapılandırma kaybı giriş ve çıkışa bağlı olarak herhangi bir şey olabilse de, denklem 3.1'de gösterildiği gibi terimi (norm kaybı olarak da adlandırılır) tasvir etmek için bir L1 kaybı kullanacağız.

$$L = |x - \hat{x}| \tag{3.1}$$

Burada $\hat{x}$ tahmin edilen çıktıyı, x ise temel gerçeği temsil eder. Kayıp fonksiyonu açık bir düzenleme terimine sahip olmadığından, modelin girdi verilerini ezberlememesini sağlamanın tek yöntemi, ağ mimarisi bölümünde darboğaz boyutunu ve gizli katman sayısını ayarlamaktır.

- **Sparse otomatik kodlayıcı**



Şekil 3.10. Tek katmanlı seyrek (sparse) otomatik kodlayıcının basit şeması.

Seyrek bir otomatik kodlayıcıda, darboğaz katmanında aslında giriş verisinden daha fazla nöron bulunur, ancak kod çözme aşamasına yalnızca az sayıda nöron (genellikle en aktif olanlar) yerleştirilir. Bu yöntem, kodlamayı depolamak için daha fazla nörona sahip olduğu için ağın giriş verileri üzerinde daha karmaşık örüntüler öğrenmesine izin verirken, yine de çok fazla nöron kullanarak girişi yeniden yapılandırmasını önler [62]. L1 düzenlemesi kullanan seyreklik faktörünün geleneksel KL ıraksamasından daha iyi sonuçlar verdiği gösterilmiştir [63].

- **Contractive otomatik kodlayıcı**

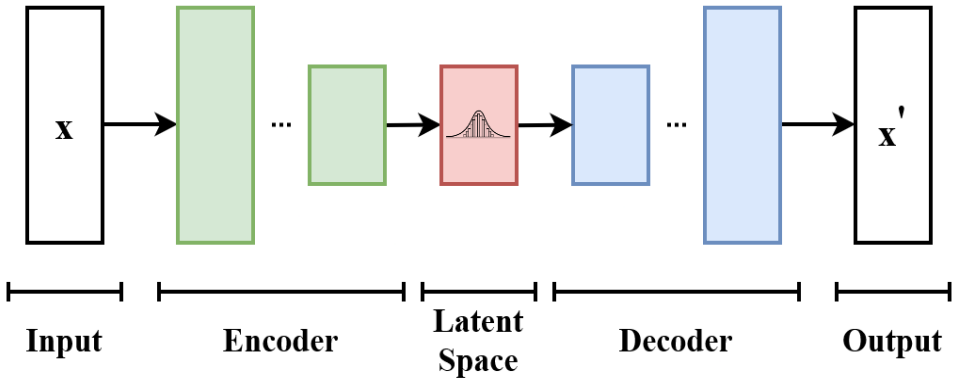
Kontraktif otomatik kodlayıcı, nöron girdilerine açık bir düzenleyici ekleyerek modeli girdi değerlerindeki küçük değişimlere karşı daha az duyarlı olmaya zorlar. Bu, esasen

nöronların verilerdeki küçük dalgalanmaları göz ardı etmesini ve yalnızca verilerdeki daha büyük, daha önemli değişikliklere tepki vermesini sağlar. Bu 'ceza' yalnızca modelin eğitimi sırasında uygulanır, bu nedenle ağı gerçekten kullanırken herhangi bir etkisi olmayacaktır [64].

- **Denoising otomatik kodlayıcı**

Otomatik bozmada, yeniden yapılandırma kriterleri değiştirilerek iyi bir temsil elde edilir. Bu, girdinin doğrudan otomatik kodlayıcıya beslenmesi yerine, önce bir tür gürültü kullanılarak bozulduğu anlamına gelir. Bu bozulmuş versiyon daha sonra otomatik kodlayıcıyı eğitmek için girdi olarak kullanılır. Performansı değerlendirmek için ağın çıktısı orijinal bozulmamış versiyonla karşılaştırılır. Ağı bu şekilde eğiterek, ağ girdiyi nasıl bozacağını öğrenir ve girdinin daha iyi bir üst düzey temsiline yol açar [65].

- **Variational otomatik kodlayıcı**



Şekil 3.11. Varyasyonel otomatik kodlayıcının temel şeması

Varyasyonel otomatik kodlayıcı, mimari benzerlikleri nedeniyle genellikle otomatik kodlayıcı modeliyle ilişkilendirilir, ancak amaç ve matematiksel formülasyonda önemli farklılıklar vardır. Varyasyonel otomatik kodlayıcı, yalnızca genel yapısının bir parçası olarak bir sinir ağı gerektiren olasılıksal bir üretken modeldir. Sinir ağı bileşenleri genellikle sırasıyla birinci ve ikinci bileşenler için kodlayıcı ve kod çözücü olarak adlandırılır. İlk sinir ağı, girdi değişkenlerini varyasyonel dağılım parametrelerine karşılık gelen gizli uzaya eşler. Bu şekilde kodlayıcı, hepsi aynı dağılımdan gelen birkaç farklı örnek üretebilir.

Kod çözücü ise veri noktaları üretmek veya oluşturmak için gizli uzaydan girdi uzayına eşleme yaparak tam tersi bir işleve sahiptir. İki ağ genellikle yeniden parametrelendirme hileleri kullanılarak birlikte eğitilir, ancak gürültü modelinin varyansı ayrı olarak öğrenilebilir. Bu tür bir model başlangıçta denetimsiz öğrenme için tasarlanmış olsa da, etkinliği yarı denetimli öğrenme [66, 67] ve denetimli öğrenme [68] için kanıtlanmıştır.

3.2.2.3. Otomatik kodlayıcı uygulamaları

Otomatik kodlayıcı mimarisi birçok alanda uygulanabilir, işte otomatik kodlayıcıların kullanıldığı bazı alanlar. İlk olarak, otoenkoder mimarisi anomali tespiti alanında kullanılmaktadır. [69] [70]. Daha önce açıklanan bazı kısıtlamalar altında eğitim verilerindeki en göze çarpan özellikleri çoğaltmayı öğrenerek, modelin en sık gözlemlenen özellikleri aslına sadık bir şekilde yeniden üretmeyi öğrenmesi teşvik edilir. Anormalliklerle karşılaştığında, model yeniden yapılandırma performansını düşürecektir. Çoğu durumda, otomatik kodlayıcıyı eğitmek için yalnızca normal örneklere sahip veriler kullanılır; diğer durumlarda, anormalliklerin sıklığı gözlem kümesine kıyasla küçüktür, bu nedenle öğrenilen temsile katkıları ihmal edilebilir düzeydedir. Eğitimden sonra, otomatik kodlayıcı "normal" verileri doğru bir şekilde yeniden yapılandırırken, alışılmadık anormal verilerle bunu yapamaz. [71]. Yeniden yapılandırma hatası (orijinal veri ile düşük boyutlu yeniden yapılandırması arasındaki hata) anomalileri tespit etmek için anomali puanı olarak kullanılır.

Ayrıca, otomatik kodlayıcılar görüntü işlemede de çok kullanışlıdır. Bir örnek, oto kodlayıcının diğer yaklaşımlardan daha iyi performans gösterdiği ve JPEG 2000'e karşı rekabetçi olduğu kanıtlanan kayıplı görüntü sıkıştırmada bulunabilir [72]. Görüntü ön işlemede otomatik kodlayıcıların bir başka yararlı uygulaması da görüntü denoising'dir [73]. Otomatik kodlayıcılar, görüntü denoising [74] ve süper çözünürlük için kullanıldıkları tıbbi görüntüleme gibi daha zorlu bağlamlarda kullanılmaktadır [75]. Görüntü destekli tanı alanında, meme kanseri tespiti [76] ve Alzheimer hastalığının bilişsel gerilemesi ile MRI ile eğitilen oto kodlayıcıların gizli özellikleri arasındaki ilişkiyi modellemek için oto kodlayıcılar uygulanmıştır [77].

Otomatik kodlayıcı mimarisi, genellikle nöral makine çevirisi (NMT) olarak adlandırılan makine çevirisine de uygulanmıştır [78]. Geleneksel otomatik kodlayıcıların aksine, çıktı girdiyle aynı değil, başka bir dildedir. NMT'de metin, öğrenme prosedürüne kodlanacak bir dizi olarak ele alınırken, kod çözücü tarafında

hedef dilde bir dizi oluşturulur. Dile özgü otomatik kodlayıcılar, öğrenme prosedürüne Çince'nin ayrıştırma özellikleri gibi daha fazla dilsel özellik dahil eder [79].

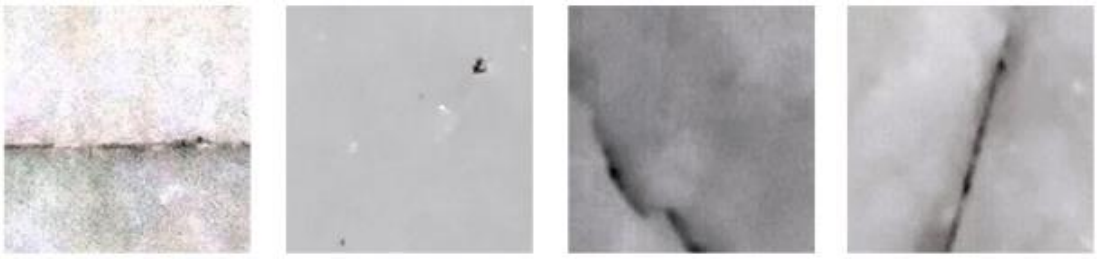
Ayrıca, popülerlik tahmini alanında, yığılmış otoenkoder (stacked autoencoder) çerçevesi sosyal medya gönderilerinin popülerliğini tahmin etmede umut verici sonuçlar sağlamaktadır [80]. Bu da çevrimiçi reklam stratejileri için çok yararlıdır.

4. DENEYSEL ÇALIŞMALAR

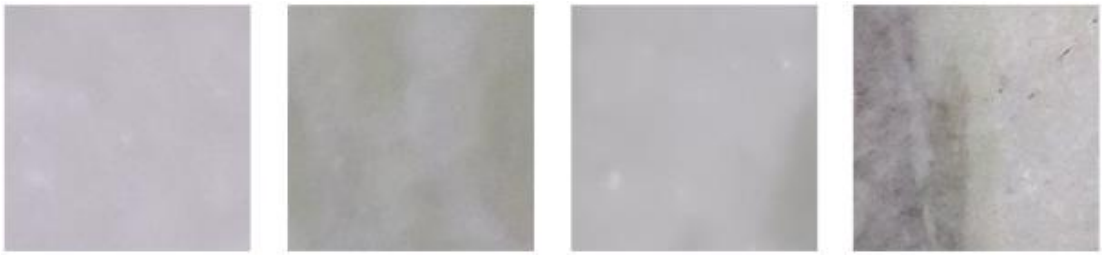
Oluşturulan mimariyi test etmek için kaggle.com'dan aldığımız mermer yüzey görüntüleri şeklindeki bir veri kümesini kullanıyoruz. Bu araştırmadaki amacımız, Autoencoder mimarisini kullanarak görüntüler aracılığıyla mermer yüzeylerindeki anormallikleri tespit etmektir.

4.1. Kullanılan Veritabanı

Kullandığımız veri kümesinde eğitim ve test olmak üzere iki sınıf bulunmaktadır. Eğitim ve test sınıfları içerisinde çatlak, nokta, iyi ve ortak olmak üzere 4 sınıf bulunmaktadır. Eğitim sınıfında toplam 2249 dosya, test sınıfında ise 688 dosya bulunmaktadır. Görüntüler 256 x 256 boyutlarındadır. Çatlak, nokta ve eklem sınıfı görüntülerinin mevcut olmaması nedeniyle, bunlar bir komut dosyası kullanılarak artırılmıştır. Bu büyütmeler kayıp türünde değildir, sadece parlaklıkta bir değişiklik ve görüntünün ters çevrilmesi (hem yatay hem de dikey) gerçekleştirilmiştir. Bu veri seti Vibhor Deshmukh tarafından Xoriant'ta derektör teknolojisi olarak oluşturulmuştur. Bu veri seti kaggle.com web sitesinde yayınlanmaktadır [81].



Şekil 4.1. Anomali (çatlak, nokta ve eklem) görüntüleri



Şekil 4.2. Normal görüntüler

4.2. Uygulama Detayları

Oluşturduğumuz modeli eğitmek için 30 epoch sayısı, 50 batch boyutu, adam optimizer ve kayıp fonksiyonu olarak 'mean_squared_error' kullandık. tüm bu deneysel çalışmalar intel işlemci i5-8250U, 16 GB RAM, 9 GB GPU NVIDIA GeForce MX130 sistem özelliklerine sahip bir dizüstü bilgisayarda gerçekleştirilmiştir. Tahmin edilen sonuçların matris tablosunu oluşturmak ve görüntülemek için Şekil 4.2.1'de gösterildiği gibi confusion_matrix ve seaborn modülünü kullandık, ardından bu çalışmada doğruluğu hesaplamak için Denklem 4.1'de gösterilen formülle scikit-learn'den accuracy_score modülünü kullandık. Ayrıca, test edilen görüntüdeki anormallikleri tespit etmek için 'yeniden yapılandırma hata eşiği' (*reconstruction error threshold*) adı verilen bir parametre oluşturduk.

$$Accuracy = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} \tag{4.1}$$

True Label	True Positive	False Positive
	False Negative	True Negative
	Prediction label	

Şekil 4.3. Confusion Matrix

4.2.1. Dizinden veri oluşturma

Başlangıç aşaması için, önerilen modelin giriş verileri için kullanılacak görüntü verilerinin yüklenmesi gerekmektedir. Bir klasörden görüntü verilerini yeniden oluşturmak için Keras küthüphane'deki Image Data Generator fonksiyonu kullanılır. Bu fonksiyon ile ne tür bir jeneratör / büyütme yapılabileceğini de belirleyebiliriz. Bu araştırmada, eğitme üreteci (generator) ve doğrulama üreteci (generator) olmak üzere 2 üreteç oluşturuyoruz ve aşağıdaki Şekil 4.4'te görüldüğü gibi 0 ile 1 arasındaki tüm piksel değerlerini bastırmak için giriş verilerini 255'e bölerek yeniden ölçeklendiriyoruz.

```
datagen = ImageDataGenerator(rescale=1./255)

train_generator = datagen.flow_from_directory(
    'good_training',
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='input'
    )

validation_generator = datagen.flow_from_directory(
    'good_val',
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='input'
    )
```

Şekil 4.4. ImageDataGenerator fonksiyonu ile veri yükleme

Ayrıca, giriş verileri için başlangıçtaki boyutu 128 piksel olarak ayarladık ve batch_size için her eğitim adımında 50 görüntü belirledik.

4.2.2. Önerilen modelin oluşturulması ve eğitilmesi

Bu aşamada, havuzlama katmanlarının yanı sıra birkaç konvolüsyon katmanlarına sahip bir otomatık kodlayıcı mimarisi oluşturduk. İlk aşamada 64, 32, 16 ve 8 boyutlarında 4 konvolüsyon katmanı ve 4 havuzlama katmanı oluşturduk ancak düşük sonuçlar elde ettik. Bunun en büyük nedeni, kod çözücü bölümünün girişi olacak olan önyükleme / kod bölümünde çok küçük piksellerin alınması, böylece çok az bilgi elde edilmesi ve giriş verilerinden çok fazla önemli bilginin boşa harcanmasıdır. Bir sonraki aşamada daha iyi sonuçlar elde etmek için, kodlayıcı kısmı için 64, 32, 16 ve MaxPooling2D katman boyutlarına sahip 3 konvolüsyon ve 3 havuzlama katmanına sahip bir mimari oluşturuyoruz ve kod çözücü kısmı için tablo 4.1'de gösterildiği gibi konvolüsyon katman boyutu 16, 32 ve 64'ten oluşuyor ve havuzlama katmanı için Upsampling2D katmanını kullanıyor.

Tablo 4.1. Önerilen otomatik kodlayıcı mimarisi

Layer (Type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 128, 128, 64)	1792
max_pooling2d_3 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_8 (Conv2D)	(None, 64, 64, 32)	18464
max_pooling2d_4 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_9 (Conv2D)	(None, 32, 32, 16)	4624
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 16)	0
conv2d_10 (Conv2D)	(None, 16, 16, 16)	2320
up_sampling2d_3 (UpSampling2D)	(None, 32, 32, 16)	0
conv2d_11 (Conv2D)	(None, 32, 32, 32)	4640
up_sampling2d_4 (UpSampling2D)	(None, 64, 64, 32)	0
conv2d_12 (Conv2D)	(None, 64, 64, 64)	18496
up_sampling2d_5 (UpSampling2D)	(None, 128, 128, 64)	0
conv2d_13 (Conv2D)	(None, 128, 128, 3)	1731

Modeli oluşturduktan sonra, oluşturulan modeli daha önce oluşturulmuş girdi verileriyle eğitiyoruz.

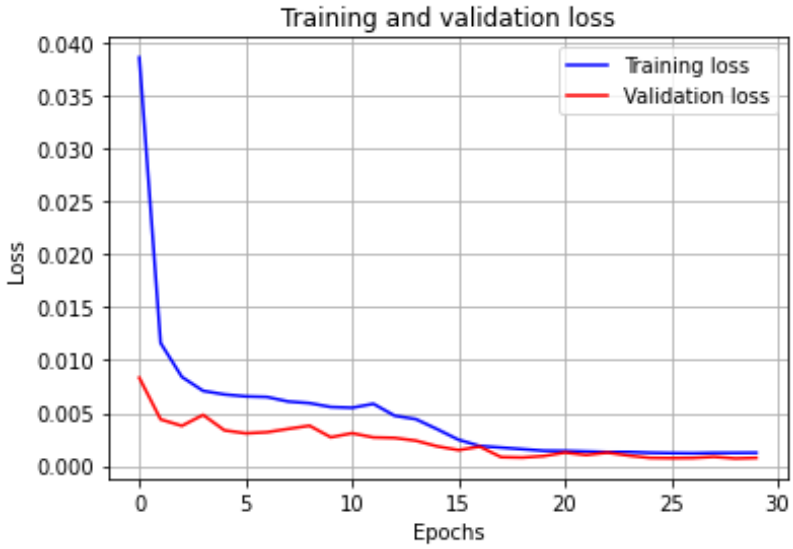
```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
csv_logger = CSVLogger('training4.log', separator=',' , append=False)

model.fit(
        train_generator,
        epochs=30,
        validation_data=validation_generator,
        callbacks = [csv_logger])
```

Şekil 4.5. Görüntü verilerinin eğitilmesi

Yukarıda şekil 4.6.'de görüldüğü gibi, eğitilecek veriler için train_generator ve verilerin doğrulanması için validation_generator olmak üzere daha önce oluşturulmuş 2 jeneratör ile verileri 30 epoch kullanarak eğitiyoruz. Ayrıca, optimize edici (*optimizer*) için adam optimizer, kayıp fonksiyonu için denklem 4.2.'te tanımlanan 'mean_squared_error' ve metrikler için 'accuracy' dahil olmak üzere giriş verilerini eğitmek için kullanılacak hiper parametreleri de ayarlıyoruz.

$$MSE = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 \tag{4.2}$$



Şekil 4.6. Kayıp fonksiyonuna dayalı eğitim sonuçları

Yukarıdaki resimden, kayıp değerinin azaldığı görülebilir, bu da yeniden yapılandırma sonuçlarının giriş verilerine yakın veya neredeyse aynı olduğu anlamına gelir.

4.2.3. Eşik (*threshold*) değerinin belirlenmesi

Bu aşamada, anomali tespiti için parametre olarak kullanılacak eşik değerini belirlemek için birkaç deney gerçekleştirdik. İlk deneyde, model.predict fonksiyonu kullanılarak oluşturulan model ile her bir girdi görüntüsünü tahmin ettikten sonra model.evaluate fonksiyonunu kullanarak yeniden yapılandırma (*reconstruction*) hata değerini bulduk ve ardından hesaplanan tüm verilerin yeniden yapılandırma hata değerinin ortalama değerini ve yeniden yapılandırma hata değerinin standart sapma (*deviation*) değerini bulduk. Elde edildikten sonra, aşağıdaki Şekil 4.8'de gösterildiği gibi yeniden yapılandırma hatasının ortalama değeri ile yeniden yapılandırma hatasının standart sapma değeri toplanır ve sonuçu eşik değerini olarak kullanılır.

```
normal_batch = train_generator.next()[0]

def recon_error(batch_images):
    recon_list=[]

    for im in range(0, batch_images.shape[0]-1):
        img = batch_images[im]
        img = img[np.newaxis, :,:,:]
        recon = simpanan_model.predict([[img]])
        recon_err = simpanan_model.evaluate([recon],[[img]], batch_size = 1)[0]
        recon_list.append(recon_err)


    average_recon_err = np.mean(np.array(recon_list))
    stdev_recon_err = np.std(np.array(recon_list))

    recon_threshold = average_recon_err + stdev_recon_err

    return average_recon_err, stdev_recon_err, recon_threshold

normal_value = recon_error(normal_batch)
```

Şekil 4.7. Birinci Deneme Eşiğinin Belirlenmesi

Ancak ne yazık ki ilk deneyde elde edilen eşik değeri ile 84% gibi çok iyi olmayan bir doğruluk sonucu elde etti, bu nedenle aşağıdaki Şekil 4.9'de gösterildiği gibi doğrulama kaybı (*valıdation loss*) değerinin ortalama değerini alarak ikinci bir deney yaptık. Ve mermer yüzey görüntü verilerindeki anormallikleri tespit etmek için eşik değeri olarak kullanıldıktan sonra, doğruluk sonuçları ilk deneyden daha iyi cıktı, yani 88% elde edildi.

```
loss = log_data['loss']
val_loss = log_data['val_loss']
plt.grid()
plt.plot(log_data['loss'], 'b', label='Training loss')
plt.plot(log_data['val_loss'], 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


threshold = np.mean(val_loss)
print(threshold)
```

Şekil 4.8. İkinci Deneme Eşiğinin Belirlenmesi

4.2.4. Anolami tespit algoritması oluşturma

Bu aşamada anomalileri tespit etmek için bir algotirma oluşturuyoruz. Kavram (Konsept) oldukça basittir, önce tespit edilecek görüntü verilerini hazırlarız, ardından kullanacağımız görüntü verilerinin adresi olarak bir yol (*path*) oluştururuz. Bundan

sonra, aşağıdaki Şekil 4.10'da gösterildiği gibi önceden tanımlanmış eşik (*threshold*) parametreleri ile tüm görüntüleri tespit etmek için bir döngü (*looping*) oluşturuyoruz.

```
import os

test_path = 'normal_test_image\images'
anomaly_path = 'anomaly_test\images'

test_name = os.listdir(test_path)
dnames = os.listdir(anomaly_path)


data_list = []
for i in range(len(dnames)):
        print(dnames[i])
        recon_err_threshold = threshold

        file_path = os.path.join(anomaly_path, dnames[i])
        img = Image.open(file_path)
        img = np.array(img.resize((128,128), Image.LANCZOS))
        img = img / 225
        img = img[np.newaxis, :,:,:]

        reconstruction = simpanan_model.predict([[img]])
        reconstruction_error = simpanan_model.evaluate([reconstruction], [[img]], batch_size = 1)[0]

        if reconstruction_error > recon_err_threshold:
            data_list.append(1)
        else:
            data_list.append(0)
```

Şekil 4.9. Anomali tespit algoritması

4.2.5. Doğruluk değerini bulma

Doğruluk değerini elde etmek için scikit-learn kütüphanesinden iki ana parametre gerektiren accuracy_score fonksiyonunu kullandık, yani temel gerçek (doğru) etiketler olarak dizi (*array*) veri tipiyle y_true ve ayrıca tahmin edilen etiketler olarak dizi veri tipiyle y_pred [82].

```
data_arr = np.array(data_list)


correct_label = []

for x in range(len(data_list)):
    if x < len(test_name):
        correct_label.append(0)
    else:
        correct_label.append(1)

correct_label_arr = np.array(correct_label)
```

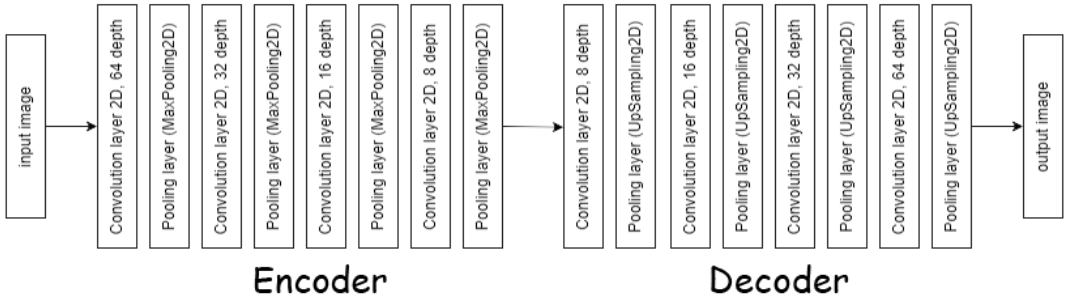
Şekil 4.10. Doğru etiketleri ve tahmin edilen etiketleri oluşturma

Yukarıdaki Şekil 4.10'da liste veri tipini np.array fonksiyonunu kullanarak dizi veri tipine dönüştürdük, bunun nedeni kullanılacak olan accuracy_score fonksiyonunun liste veri tiplerini işleyememesidir. Ayrıca, giriş verisi kadar bir doğru etiketi veya y_true oluştururuz ve tahmin etiketi veya y_pred için yukarıdaki Şekil 4.9'da

gösterildiği gibi yapılan algılama algoritmasının çıkış değerini kullanır. Bundan sonra, iki parametre accuracy_score fonksiyonu kullanılarak işlenir. Yapılan deneylerin sonuçlarını görmek için bir sonraki bölümde ayrıntılı olarak ele alınacaktır.

4.3. Deneme Sonuçları

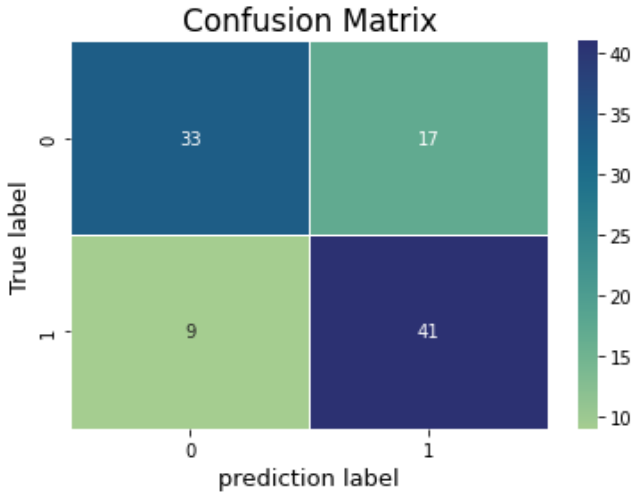
Bu araştırmada, en iyi doğruluk sonuçlarını elde etmek için çeşitli testler gerçekleştirilmiştir. İlk testte, kodlayıcı ve kod çözücü bölümlerinde 64, 32, 16, 8 ve tersi olmak üzere 4 konvolüsyon katmanı ve Şekil 4.11'de gösterildiği gibi kodlayıcı bölümünde MaxPooling2D katmanını ve kod çözücü bölümünde UpSampling2D katmanını kullanarak 4 havuzlama katmanı kullandık.



Şekil 4.11. Birinci denemede kullanılan otomatik kodlayıcı mimarisi

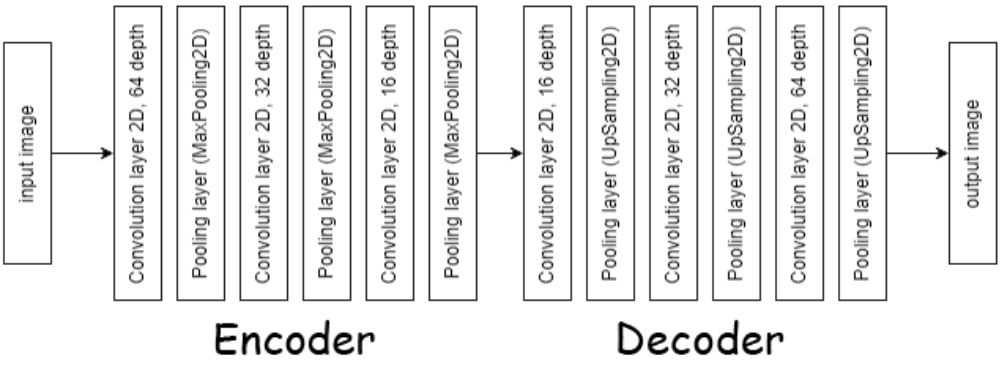
Konvolüsyon katmanları, daha sonra özellikleri eşlemek için kullanılan bir filtre veya çekirdek kullanılarak giriş verileri üzerinde çalıştırılır. Konvolüsyon işlemi, filtrenin giriş üzerinde kaydırılmasıyla gerçekleştirilir. Her konumda, matris çarpımı gerçekleştirilir ve sonuçlar bir özellik haritasında toplanır. konvolüsyon işleminin çıktısı bir aktivasyon fonksiyonundan geçirilecektir. bu çalışmada, çıktıyı doğrusal olmayan (non-linear) hale getirmek için aktivasyon fonksiyonu olarak relu kullanılmıştır.

Bundan sonra, havuzlama katmanına girilir, bu havuzlama katmanı özellik haritasının boyutunu azaltmayı amaçlar ve yalnızca giriş verilerinden önemli ve anlamlı bilgileri depolar. Bundan sonra, 300 görüntü verisi, oluşturulan otoenkoder modelini eğitmek için kullanılır. Daha sonra eğitilen model, anormallikleri tespit etmek için eşik değeri olarak kullanılacak olan yeniden yapılandırma hata değerini bulmak için kullanılır. Daha sonra algoritmayı test etmek için 50 anormal görüntü ve 50 normal görüntü hazırladık ve aşağıdaki Şekil 4.10'da gösterildiği gibi toplam 74 doğru veri ve toplam 26 yanlış veri ile %74'lük bir başarı oranı elde ettik.

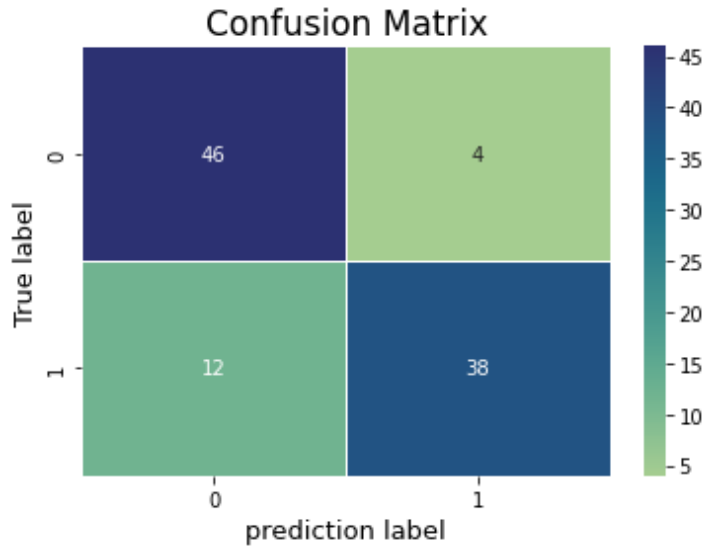


Şekil 4.12. Karışıklık matrisi üzerinde ilk deneme sonucu

Daha sonra, ikinci denemede, aşağıdaki şekil 4.13.te gösterildiği gibi sayıyı kodlayıcı kısmı için 64, 32 ve 16 boyutlarında 3 konvolüsyon katmanına ve 3 havuzlama katmanına ve kod çözücü kısmı için 16, 32, 64 boyutlarında 3 konvolüsyon katmanına ve 3 havuzlama katmanına düşürdük, bunun amacı alınan giriş verileriyle ilgili bilgilerin çok az olmaması ve giriş verileriyle ilgili bilgilerin çok fazla boşa harcanmaması ve eğitim sonuçlarını iyileştirmek ve daha iyi bir yeniden yapılandırma hata eşiği elde etmek için eğitilen görüntü verilerinin sayısını 400 görüntüye çıkarmaktır. Daha önce olduğu gibi aynı test verileriyle, yani 50 anormal görüntü ve 50 normal görüntüyle test edildikten sonra, aşağıdaki Şekil 4.14'de gösterildiği gibi 84 doğru veri ve 16 yanlış veri ile 84%'lük bir doğruluk elde edilmiştir.

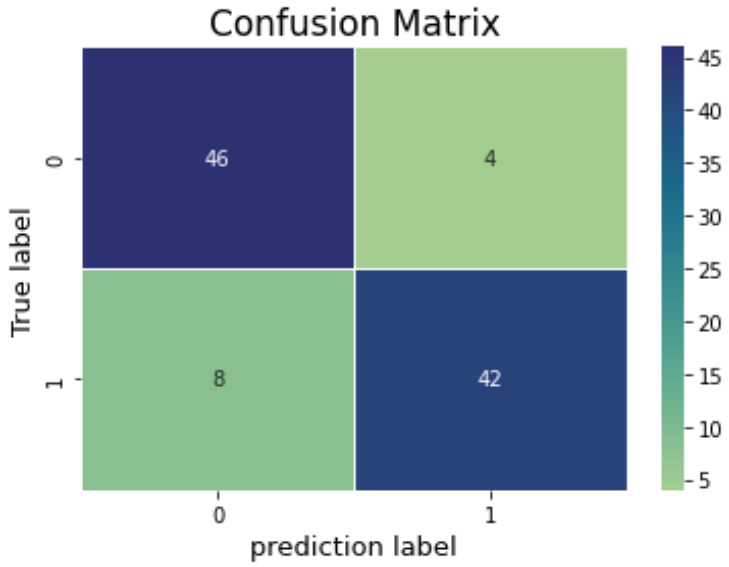


Şekil 4.13. İkinci denemede kullanılan otomatik kodlayıcı mimarisi



Şekil 4.14. Karışıklık matrisi üzerinde ikinci deneme sonucu

Daha sonra üçüncü denemede katman sayısı ikinci denemeyle aynı olacak şekilde, yani yükarıdaki şekil 4.13'de gösterildiği gibi 3 konvolüsyon katmanı ve 3 havuzlama katmanı kullanarak 50 normal görüntü verisi ve 50 anormal görüntü verisi kullanarak yaptık. Ancak bu üçüncü denemede, önceki 2 denemeden farklı bir eşik değeri kullanılmaktadır, yani doğrulama kaybı değerinin ortalama değeri alınmakta ve bu değer daha sonra eşik değeri olarak kullanılmaktadır ve aşağıdaki Şekil 4.15'de gösterildiği gibi 88 görüntü kadar doğru veri ve 12 görüntü kadar yanlış veri sayısı ile 88%'lik bir doğruluk değeri ile önceki 2 denemeden çok daha yüksek sonuçları elde ettik.



Şekil 4.15. Karışıklık matrisi üzerinde üçüncü deneme sonucu

Yapılan deneme sonuçlarını göstermek için karışıklık matrisini (*confusion matrix*) kullanıyoruz [83, 84]. Karışıklık matrisi öğrenme kalitesini gösterir, satırlar tahmin edilen sınıfı gösterirken sütunlar örneğin gerçek sınıfını gösterir. Sağ alt sütun doğru tahmin edilen anomali görüntülerinin sayısını, sol alt sütun ise yanlış tahmin edilen anomali görüntülerinin sayısını göstermektedir. Şekil 4.3'te gösterildiği gibi sağ üst sütun yanlış tahmin edilen iyi görüntülerin sayısını, sol üst sütun ise doğru tahmin edilen iyi görüntülerin sayısını göstermektedir.

Doğruluk değerine ek olarak, yapılan denemelerden hassasiyet (precision), hatırlama (recall) ve f1 değerlerini de elde edilmiştir. iki farklı model ve üç farklı eşik değeriyle 3 deneme yapıldıktan sonra, aşağıdaki tablo 4.4'te gösterildiği gibi üçüncü denemeyi 88% doğruluk değeriyle en yüksek sonuçları verdiği görülmüştür.

Tablo 4.2. Deneme Sonuçları

Veriset (Dataset)	**Deneme**	**Accuracy**	**Precision Score**	**Recall Score**	**F1 Score**
Marble Surface Anomaly Detection	1.	74%	88%	70%	80%
Marble Surface Anomaly Detection	2.	84%	90%	76%	83%
Marble Surface Anomaly Detection	**3.**	**88%**	**91%**	**84%**	**87%**

5. SONUÇ VE ÖNERİLER

Bu araştırmada, hazırlanan görüntü verilerini eğitmek ve bir anormallik tespit algoritması oluşturmak için yapay sinir ağı türlerden biri otomatik kodlayıcı mimarisini kullanıyoruz. Bu araştırmanın amacı mermer yüzeyinde anormalliklerin olup olmadığını görüntü şeklindeki girdi verileri aracılığıyla tespit etmektir.

Bu araştırmada doğru eşiğin bulunması çok önemlidir çünkü eşik anolami tespitinde parametre olarak kullanılacaktır. Bu nedenle, maksimum sonuçları elde etmek için katman sayısını, eğitim verilerinin miktarını ve ayrıca birkaç hiper parametreyi (yineleme sayısı, optimize edici, giriş boyutu ve parti (*batch*) boyutu) belirlemek için birkaç testler yapmak gerekmektedir.

Yukarıda tartışıldığı gibi, otomatik kodlayıcı mimarisi için kodlayıcı bileşeninde 6 katman, 3 evrişim katmanı ve 3 havuzlama katmanı yaptık, benzer şekilde kod çözücü katmanı için de 6 katman, 3 evrişim katmanı ve 3 havuzlama katmanı yaptık. Bunun dışında giriş boyutunu 128 piksele, yineleme (*epoch*) sayısını 30'a, parti (*batch*) boyutunu 50 görüntüye ve Adam optimizer'ı kullanarak ayarladık. Oluşturulan algoritmayı test etmek için ise "mermer yüzeyi anomali tespiti" veri seti kullanılmış ve %88 doğruluk oranı elde edilmiştir.

Daha sonraki araştırmalarda, anomali tespit sonuçlarına anomalilerin ayrıntılı açıklamaları (çatlaklar, eklemler, noktalar vb.) eklenerek bu araştırmanın geliştirilmesi gerçekleştirilebilir. Bunun dışında anormallikleri tespit etmek için kullanılan parametre artırılarak da geliştirme yapılabilir.

KAYNAKLAR

[1] B. team, «what-marble-is-used-for,» Stonemason Supplies Pty Ltd, [Çevrimiçi]. Available: https://cmpstone.com.au/. [Erişildi: 3 November 2023].

[2] H. Uzen, M. Turkoglu ve D. Hanbay, «Texture defect classification with multiple pooling and filter ensemble based on deep neural network,» Expert System with Applications, August 2021.

[3] D. Zhang, K. Song, J. Xu, Y. He, M. Niu ve Y. Yan, «MCnet: Multiple Contect Information Segmentation Network of No-Service Rain Surface Defects,» IEEE Transaction on Instrumentation and Meansurement, 2021.

[4] H. Dong, K. Song, Y. He, J. Xu, Y. Yan ve Q. Meng, «PGA-Net: Pyramid Feature Fusion and Global Context Attention Network for Automated Surface Defect Detection,» IEEE Transaction on Industrial Informatics, December 2020.

[5] K. Hanbay, M. F. Talu ve Ö. F. Özgüven, «Fabric defect detection system and methods,» A systematic literature review, December 2016.

[6] T. Özseven, «Surface Defect Detection and Quantification with Image Processing Methods,» March 2019.

[7] D. Tsai, P. Lin ve C. Lu, «An independent component analysis-based filter design for defect detection in low-contrast surface images,» Pattern Recognit, p. 1679–1694, 2006.

[8] A. Latif-Amet, A. Ertüzün ve A. Erçil, «An efficient method for texture defect detection: sub-band domain co-occurrence matrices,» Image Vision Computer, p. 543–553, 2000.

[9] H. Jia, Y. L. Murphey, J. Shi ve T.-S. Chang, «An intelligent real-time vision system for surface defect detection,» in Pattern Recognition, p. 239–242, 2004.

[10] M. A. Coulthard, «Image processing for automatic surface defect detection,» in Image Processing and its Applications, p. 192–196, 1989.

[11] S. H. Bhandari, S. M. Deshpande ve S. M. Deshpande, «A Simple Approach to Surface Defect Detection,» IEEE Reg, p. 8–10, 2008.

[12] A. Kumar ve G. K. Pang, «Defect detection in textured materials using Gabor filters,» IEEE Trans. Ind. Appl, p. 425–440, 2002.

[13] D. Racki, D. Tomazevic ve D. Skocaj, «A compact convolutional neural network for textured surface anomaly detection,» IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1331-1339, March 2018.

[14] M. S. Minhas, «Anomaly Detection in Textured Surfaces,» 2016.

[15] H. ÜZEN, M. TÜRKOĞLU ve D. HANBAY, «Result Weighting-Based Resnet Feature Pyramid Network Architecture for Surface Defect Detection,» 2021.

[16] B. Staara, M. Lütjena ve M. Freitag, «Anomaly detection with convolutional neural networks for industrial surface inspection,» Procedia CIRP, 2019.

[17] O. Rippel, M. Müller ve D. Merhof, «GAN-based Defect Synthesis for Anomaly Detection in Fabrics,» IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020.

[18] D. Tabernik, S. Šela, J. Skvarc ve D. Skocaj, «Segmentation-based deep-learning approach for surface-defect detection,» Journal of Intelligent Manufacturing, 2020.

[19] T. E. o. E. Britannica, «Science & Tech,» 29 October 2023. [Çevrimiçi]. Available: www.britannica.com. [Erişildi: 19 December 2023].

[20] P. Kearey, Dictionary of Geology, London and New York: Penguin Group, 2001.

[21] G. Marras, G. Carcangiu, P. Meloni ve N. Careddu, «Circular economy in marble industry: From stone scraps to sustainable water-based paints,» Construction and Building Materials., 28 March 2022.

[22] «Geology - rocks and minerals,» [Çevrimiçi]. Available: https://rocksminerals.flexiblelearning.auckland.ac.nz. [Erişildi: 24 12 2023].

[23] «marble-global-market-report,» The Business Research Company, January 2023. [Çevrimiçi]. Available: https://www.thebusinessresearchcompany.com/. [Erişildi: 2 November 2023].

[24] F. Y. Edgeworth, «On discordant observations,» Philosophical Magazine, pp. 364-375, 1887.

[25] R. Fujimaki, T. Yairi ve K. Machida, «An approach to spacecraft anomaly detection problem using kernel feature space,» In Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining., pp. 401-410, 2005.

[26] C. Spence, L. Parra ve P. Sajda, «Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model,» In Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, 2001.

[27] V. Kumar, «Parallel and distributed computing for cybersecurity,» Distributed Systems Online, 2005.

[28] E. Aleskerov, B. Freisleben ve B. Rao, «Cardwatch: A neural network based database mining system for credit card fraud detection,» In Proceedings of IEEE Computational Intelligence for Financial Engineering, pp. 220-226, 1997.

[29] R. Kemmerer ve G. Vigna, «Intrusion detection: a brief history and overview,» pp. supl27 - supl30, April 2002.

[30] R. Chalapathy ve S. Chawla, «Deep learning for anomaly detection: A survey,» CoRR, 2019.

[31] V. Chandola, A. Banerjee ve V. Kumar, «Anomaly detection: A survey,» ACM Computing Surveys (CSUR), 2009.

[32] V. H. Pham ve B. R. Lee, «An image segmentation approach for fruit defect detection using k-means clustering and graph-based algorithm,» Vietnam Journal of Computer Science, February 2015.

[33] T. Schlegl, P. Seebock, S. M. Waldstein, U. Schmidt-Erfurth ve G. Langs, «Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,» Information Processing in Medical Imaging, 2017.

[34] H. Zenati, M. Romain, C. S. Foo, B. Lecouat ve V. R. Chandrasekhar, «Adversarially learned anomaly detection,» IEEE International Conference on Data Mining(ICDM), pp. 727-736, 2018.

[35] S. Youkachen, M. Ruchanurucks, T. Phatrapomnant ve H. Kaneko, «Defect segmentation of hot-rolled steel strip surface by using convolutional auto-encoder and conventional image processing,» 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), pp. 1-5, March 2019.

[36] M. V. Joshi, R. C. Agarwal ve V. Kumar, «Mining needle in a haystack: classifying rare classes via two-phase rule induction,» In Proceedings of the 2001 ACM SIGMOD international conference on Management of data, pp. 91-102, 2001.

[37] M. V. Joshi, R. C. Agarwal ve V. Kumar, «Predicting rare classes: can boosting make any weak learner strong?,» In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, no. 2002, pp. 297-306, 2002.

[38] N. V. Chawla, N. Japkowicz ve A. Kotcz, «Editorial: special issue on learning from imbalanced data sets,» SIGKDD Explorations, pp. 1-6, 2004.

[39] C. Phua, D. Alahakoon ve V. Lee, «Minority report in fraud detection: classification of skewed data,» SIGKDD Explorer Newsletter, pp. 50-59, 2004.

[40] N. Abe, B. Zadrozny ve J. Langford, «Outlier detection by active learning,» In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 504-509, 2006.

[41] I. Steinwart, D. Hush ve C. Scovel, «A classi¯cation framework for anomaly detection,» Journal of Machine Learning Research, pp. 211-232, 2005.

[42] J. Theiler ve D. M. Cai, «Resampling approach for anomaly detection in multispectral images,» In Proceedings of SPIE 5093, pp. 230-240, 2003.

[43] D. E. Denning, «An Intrusion-Detection Model,» IEEE Transactions on Software Engineering, p. 222–232, 1987.

[44] H. S. Teng, K. Chen ve S. C. Lu, «Adaptive real-time anomaly detection using inductively generated sequential patterns,» IEEE Computer Society Symposium on Research in Security and Privacy, p. 278–284, 1990.

[45] M. R. Smith ve T. Martinez, «Improving classification accuracy by identifying and removing instances that should be misclassified,» The 2011 International Joint Conference on Neural Networks, 2011.

[46] M. Qasim ve E. Verdu, «Video anomaly detection system using deep convolutional and recurrent models,» Results in Engineering, 1 June 2023.

[47] T. Zhang, A. Chowdhery, P. (. Bahl, K. Jamieson ve S. Banerjee, «The Design and Implementation of a Wireless Video Surveillance System,» Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, p. 426–438, 7 September 2015.

[48] A. Chatterjee ve B. S. Ahmed, «IoT anomaly detection methods and applications: A survey,» Internet of Things, August 2022.

[49] S. Garg, K. Kaur, S. Batra, G. Kaddoum, N. Kumar ve A. Boukerche, «A multi-stage anomaly detection scheme for augmenting the security in IoT-enabled applications,» Future Generation Computer Systems, p. 105–118, 1 March 2020.

[50] «Python Documentation,» Python Software Foundation, [Çevrimiçi]. Available: https://docs.python.org/3/. [Erişildi: 20 12 2023].

[51] «Developer Survey,» Stack Overflow, 2022. [Çevrimiçi]. Available: https://survey.stackoverflow.co/2022/. [Erişildi: 20 December 2023].

[52] M. Lutz, Programming Python, L. Laura ve W. Frank, Dü, United States of America: O'Reilly and Associates, 2001.

[53] «Home,» Spyder Website Contributors, [Çevrimiçi]. Available: https://www.spyder-ide.org/. [Erişildi: 20 12 2023].

[54] D. Nasution, T. H. F. Harumy, E. Haryanto, F. Fachrizal, Julham ve A. Turnip, «A classification method for prediction of qualitative properties of multivariate EEG-P300 signals,» International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT), 29-30 October 2015.

[55] C. Lim, S. Woo, A. Loh ve R. Osman, «Speech recognition using artificial neural networks,» Proceedings of the First International Conference on Web Information Systems Engineering, 6 August 2002.

[56] D. Vinita ve D. Sunil, «Hand Written Character Recognition Using Artificial Neural Network,» Advances in Computing, pp. 18-23, 2011.

[57] K. Ali, D. Bassam ve B. Samia, «Offline signature recognition using neural networks approach,» Procedia Computer Science, pp. 155-161, 22 February 2011.

[58] N. Ganesan, K. Venkatesh, M. A. Rama ve A. M. Palani, «Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data,» International Journal of Computer Applications, p. 81–97, 2010.

[59] R. Nix ve J. Zhang, «Classification of Android apps and malware using deep neural networks,» International Joint Conference on Neural Networks (IJCNN), p. 1871–1878, May 2017.

[60] S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha ve R. Khayami, «BoTShark: A Deep Learning Approach for Botnet Traffic Detection,» Cyber Threat Intelligence, Advances in Information Security, p. 137–153, 2018.

[61] M. Sewak, S. K. Sahay ve H. Rathore, «An Overview of Deep Learning Architecture of Deep Neural Networks and Autoencoders,» cilt VII, pp. 182-188, 1 january 2020.

[62] A. Makhzani ve B. Frey, «k-Sparse Autoencoders,» arXiv e-prints, December 2013.

[63] L. Zhang, Y. Lu, B. Wang, F. Li ve Z. Zhang, «Sparse Auto-encoder with Smoothed l1 Regularization,» Neural Processing Letters, June 2018.

[64] S. Rifai, X. Muller, X. Glorot, G. Mesnil, Y. Bengio ve P. Vincent, «Learning invariant features through local space contraction,» arXiv:1104.4153 [cs], April 2011.

[65] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol ve L. Bottou, «Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.,» Journal of machine learning research, 2010.

[66] M. Ehsan Abbasnejad, A. Dick ve A. van den Hengel, «Infinite Variational Autoencoder for Semi-Supervised Learning,» p. 5888–5897, 2017.

[67] W. Xu, H. Sun, C. Deng ve Y. Tan, «Variational Autoencoder for Semi-Supervised Text Classification,» Proceedings of the AAAI Conference on Artificial Intelligence, 12 February 2017.

[68] H. Kameoka, L. Li, S. Inoue ve S. Makino, «Supervised Determined Source Separation with Multichannel Variational Autoencoder,» Neural Computation, p. 1891–1914, 1 September 2019.

[69] A. Morales-Forero ve S. Bassetto, «Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis,» 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), p. 1031–1037, December 2019.

[70] M. Sakurada ve T. Yairi, «Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction,» Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, p. 4–11, December 2014.

[71] J. An ve S. Cho, «Variational Autoencoder based Anomaly Detection using Reconstruction Probability,» Special Lecture on IE, pp. 1-18, 2015.

[72] J. Balle, V. Laparra ve E. Simoncelli, «End-to-end optimized image compression,» International Conference on Learning Representations, April 2017.

[73] K. Cho, «Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images,» In International Conference on Machine Learning, pp. 432-440, February 2013.

[74] L. Gondara, «Medical Image Denoising Using Convolutional Denoising Autoencoders,» IEEE 16th International Conference on Data Mining Workshops (ICDMW), p. 241–246, December 2016.

[75] S. Tzu-Hsi, V. Sanchez, E. Hesham ve R. Nasir M., «Hybrid deep autoencoder with Curvature Gaussian for detection of various types of cells in bone marrow trephine biopsy images,» IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017), p. 1040–1043, 2017.

[76] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang ve A. Madabhushi, «Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images,» IEEE Transactions on Medical Imaging, p. 119–130, January 2016.

[77] F. J. Martinez-Murcia, A. Ortiz, J. M. Gorriz, J. Ramirez ve D. Castillo-Barnes, «Studying the Manifold Structure of Alzheimer's Disease: A Deep Learning Approach Using Convolutional Autoencoders,» IEEE Journal of Biomedical and Health Informatics, p. 17–26, 2020.

[78] K. Cho, B. v. Merrienboer, D. Bahdanau ve Y. Bengio, «On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,» 2014.

[79] L. Han ve S. Kuang, «Incorporating Chinese Radicals into Neural Machine Translation: Deeper Than Character Level,» 2018.

[80] S. De, A. Maity, V. Goel, S. Shitole ve A. Bhattacharya, «Predicting the popularity of instagram posts for a lifestyle magazine using deep learning,» 2nd IEEE International Conference on Communication Systems, Computing and IT Applications (CSCITA), p. 174–177, 2017.

[81] V. Deshmukh, «Datasets,» 2021. [Çevrimiçi]. Available: https://www.kaggle.com. [Erişildi: 2 11 2023].

[82] «sklearn.metrics.accuracy_score,» [Çevrimiçi]. Available: https://scikit-learn.org. [Erişildi: 24 12 2023].

[83] M. J. Brusco ve J. D. Cradit, «Graph Coloring, Minimum-diameter Partitioning, and the Analysis of Confusion Matrices,» October 2004.

[84] X. Deng, Q. Liu, Y. Deng ve S. Mahadevan, «An improved method to construct basic probability assignment based on the confusion matrix for classification problem,» Information Sciences, 2016.

ÖZGEÇMİŞ

Ad-Soyad : Muhammad Yahya ABDULLAH

ÖĞRENİM DURUMU:

- **Lisans :** 2020, Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Informatics Engginering

MESLEKİ DENEYİM:

- 2020 (şubat - haziran) ayları arasında NFjuara şirketinde data entry olarak çalıştım.
- 2020 (temmuz - aralık) ayları arasında Veri Merkezi ve Veri Ambarı Yönetim Görevlisi olarak çalıştım.