Teknoloji Fakültesi

MRM3048.1 –Mekatronik Sistem Tasarımı

Ödev Raporu

Omar Tarabin 170221992

Görüntü İşleme ile Saç Levha Üzerindeki Yüzey Hataları Tespiti

Bu raporda, saç levha resimleri üzerinde görüntü işleme yöntemleri uygulanarak, levhaların üzerindeki gelebilecek olan bazı hataların otomatik olarak tespit etmesi üzerine çalışma yapılacaktır. Bu çalışmada önce veri toplama yapıldıktan sonra, Python programlama dili ve TensorFlow makine öğrenimi çerçevesi kullanılarak bir CNN modeli oluşturuldu. Ardından bu modeli toplanan verilerle eğittikten sonra nihai modelin doğrulaması yapıldı ve sonuçlar analiz edildi.

1. Sac Metalde Yüzey Kusurlarinin Tespitinin Önemi

Sac metaldeki yüzey kusurlarını tespit etmenin önemi, ürün kalitesini sağlamak, maliyetleri düşürmek, müşteri memnuniyetini artırmak, güvenliği sağlamak ve düzenlemelere uymaktır. Görüntü İşleme ve özellikle CNN mimarileri gibi araçları kullanan bilgisayar teknolojileri, modern üretim süreçlerinde bu kusurları tespit etmek ve ele almak için verimli ve etkili yöntemler sunar. Bu çalışmanın önemi aşağıdaki maddelerle özetlenebilir:

- Kalite Güvencesi: Kusurların tespit edilmesi, yüksek kaliteli malzemelerin kullanılmasını sağlayarak ürün kalitesini ve güvenilirliğini artırır.
- Maliyet Azaltma: Kusurların erken tespit edilmesi malzeme ve yeniden işleme maliyetlerini azaltarak üretim verimliliğini optimize eder.
- Müşteri Memnuniyeti: Hatasız ürünler müşteri memnuniyetini ve sadakatini artırır.
- Güvenlik: Kusurların tespit edilmesi, özellikle otomotiv ve havacılık gibi kritik uygulamalarda kazaları önler.
- Uyumluluk ve Yönetmelikler: Hatasız ürünlerin yasal ve güvenlik gereksinimlerini karşılamasını sağlamak, cezalardan kaçınmak ve pazara erişimi sağlamak.

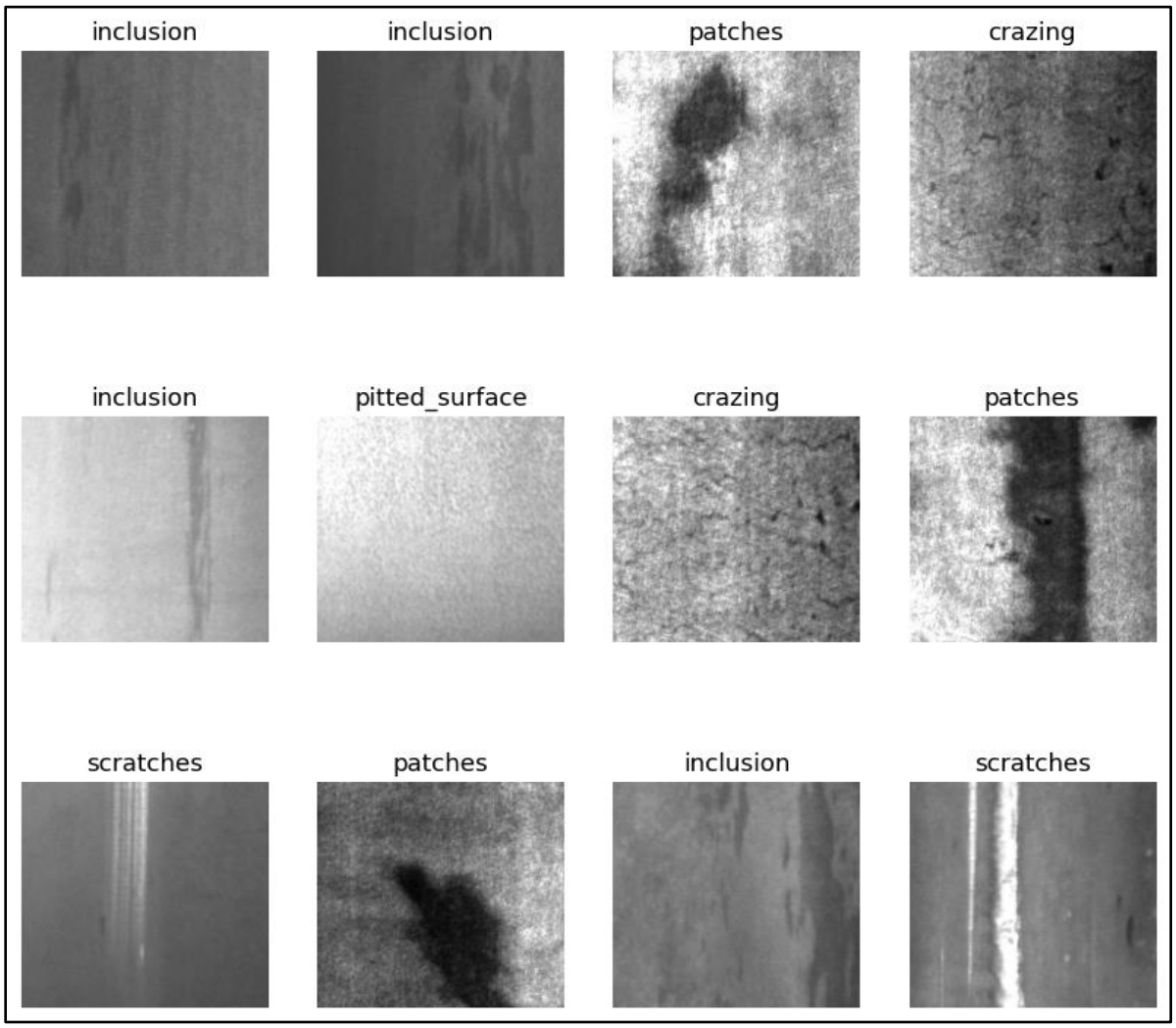
2. Yüzey Kusurları Örnekleri

Bu çalışmada NEU Metal Yüzey Kusurları veri seti kullanılmıştır. Bu veri seti, sıcak haddelenmiş çelik şeritlerin altı tipik yüzey kusur fotoğraf örneklerini içerir. Veritabanı, her altı tipik yüzey kusurundan 300 fotoğraf olmak üzere toplamda 1800 fotoğraf içerir. Bu veri kümesinin içerdiği altı farklı kusur türleri şunlardır:

- Çizikler (Scratches): Çizikler metal levhanın taşınması, nakliyesi veya işlenmesi sırasında oluşabilir. Genellikle başka bir yüzey veya nesne ile aşındırıcı temastan kaynaklanırlar.

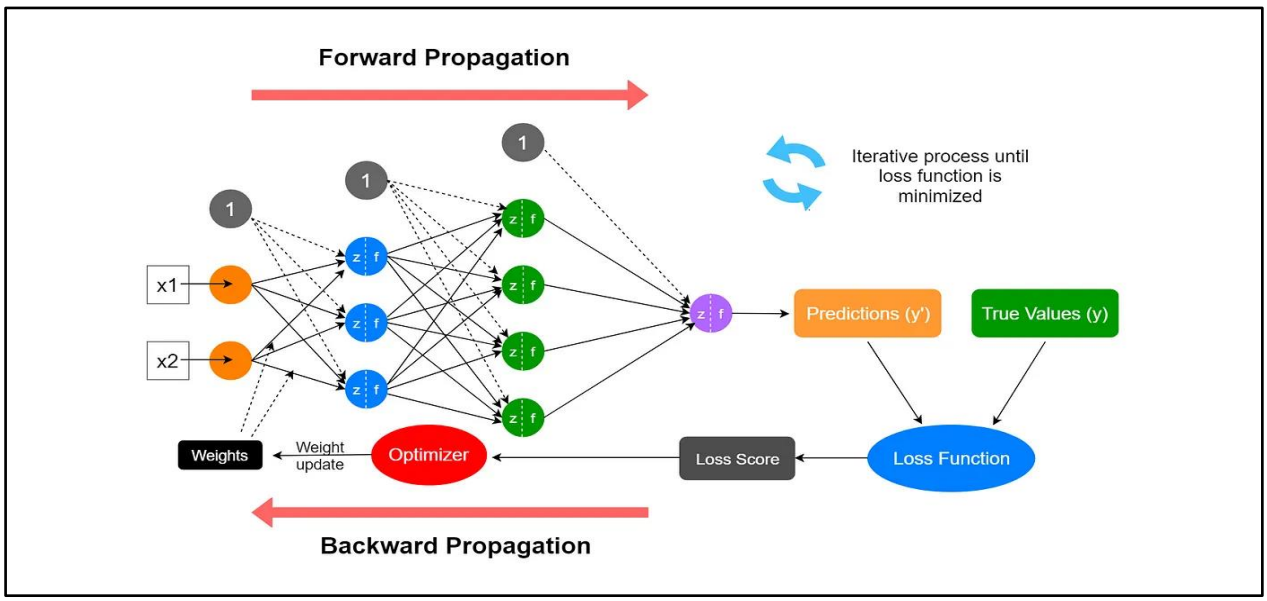
- Silindir izleri (Roller Marks): Merdane izleri, sac levhanın üretim sürecinde kullanılan merdanelerdeki kusurlar veya düzensizliklerden kaynaklanır. Bu düzensizlikler, silindirlerden geçerken metal yüzeyinde izler bırakabilir.
- Çukurlu yüzey (Pitted Surface): Sac levhanın yüzeyindeki çukurlar veya kraterler, metaldeki yabancı maddelerden veya imalat sırasında metale gömülen yabancı parçacıkların varlığından kaynaklanabilir.
- Yerel lekeler (Patches): Metalin bileşimindeki veya kalınlığındaki değişiklikler ya da işleme sırasında lokal ısıtma veya soğutma nedeniyle farklı tonlarda yerel yamalar oluşabilir.
- Çatlama (Crazing): Çatlama, sac metal yüzeyinde ince çatlakların oluşması anlamına gelir. Bu, metal üzerindeki stres veya zorlanma, sert çevre koşullarına maruz kalma veya yanlış kullanım nedeniyle meydana gelebilir.
- İnklüzyon (Inclusion): İnklüzyonlar, imalat sırasında sac metalin yüzeyine gömülen yabancı maddelerdir. Bunlar orijinal metal bileşiminin bir parçası olmayan toz, kir veya diğer kirletici parçacıkları içerebilir.

Veri içerisinde bulunan bazı görseller aşağıdaki gibidir:



3. Yapay Sinir Ağları ve CNN

Yapay sinir ağları birbirine bağlı yapay nöron denen yapıların birleşiminden oluşan bir yapıdır. Bu ağlar önce toplanan etiketlenmiş verileri kullanarak eğitilir. Eğitilmesi sırasında içindeki parametreler (ağırlıklar) her gelen veri ile Gradiant Descent ve Backpropogation algoritmalarına bağlı olarak hata eğrisini en aza indirecek şekilde güncellenir. Bu şekilde yapay sinir ağları özellikle direkt olarak modellemesi mümkün olmayan çok karmaşık yapıları modellemek için çok başarılı bir yöntemdir. Eğitim sonunda bu aşamaya ayrılmış özel bir veri grubu ile modelin doğrulaması yapılır ve eğer sonuçlar uygun görülürse model tespit için kullanılabilir.



Evrişimli Sinir Ağı (CNN), görüntü tanıma ve işleme için yaygın olarak kullanılan, evrişimli katmanlar aracılığıyla özelliklerin uzamsal hiyerarşilerini otomatik ve uyarlanabilir bir şekilde öğrenmek için tasarlanmış bir tür derin sinir ağıdır. Görüntü tanıma, sınıflandırma ve segmentasyon gibi görevler için özellikle güçlüdürler. Bir girdi olarak bir görüntü aldıktan sonra onu çeşitli katmanlardan geçirerek görüntüdeki farklı özelliklere değer verir (ağırlık verir) ve sınıflandırma yapar. Temel yapısı ve çalışma mantığı şu şekildedir:

1. Evrimsel Katmanlar: Bir CNN'in temel yapı taşı konvolüsyonel katmandır. Bir giriş görüntüsüne bir dizi filtre (kernel olarak da adlandırılır) uygular. Her filtre, görüntüdeki farklı uzamsal konumlarda kenarlar veya dokular gibi belirli özellikleri algılar. Bu işlemin çıktısı, girdideki bu özelliklerin varlığını vurgulayan bir dizi "özellik haritası "dır.

2. Aktivasyon Fonksiyonu: Tipik olarak, her bir evrişimsel katmanı ReLU (Rectified Linear Unit) gibi doğrusal olmayan bir aktivasyon fonksiyonu takip eder. Bu, modele doğrusal olmayan bir özellik kazandırarak verilerdeki daha karmaşık örüntüleri ve lineer olmayan yapıları öğrenmesini sağlar.

3. Biriktirme (Pooling) Katmanları: Bir veya daha fazla evrişimsel katmandan sonra, önemli bilgileri korurken özellik haritalarının uzamsal boyutlarını azaltmak için genellikle pooling katmanları uygulanır. Pooling işlemleri, her bölgedeki maksimum veya ortalama değeri alarak

özellik haritalarının boyutluluğunu ve hesaplama karmaşıklığını azaltırken önemli bilgilerin korunmasına yardımcı olur.

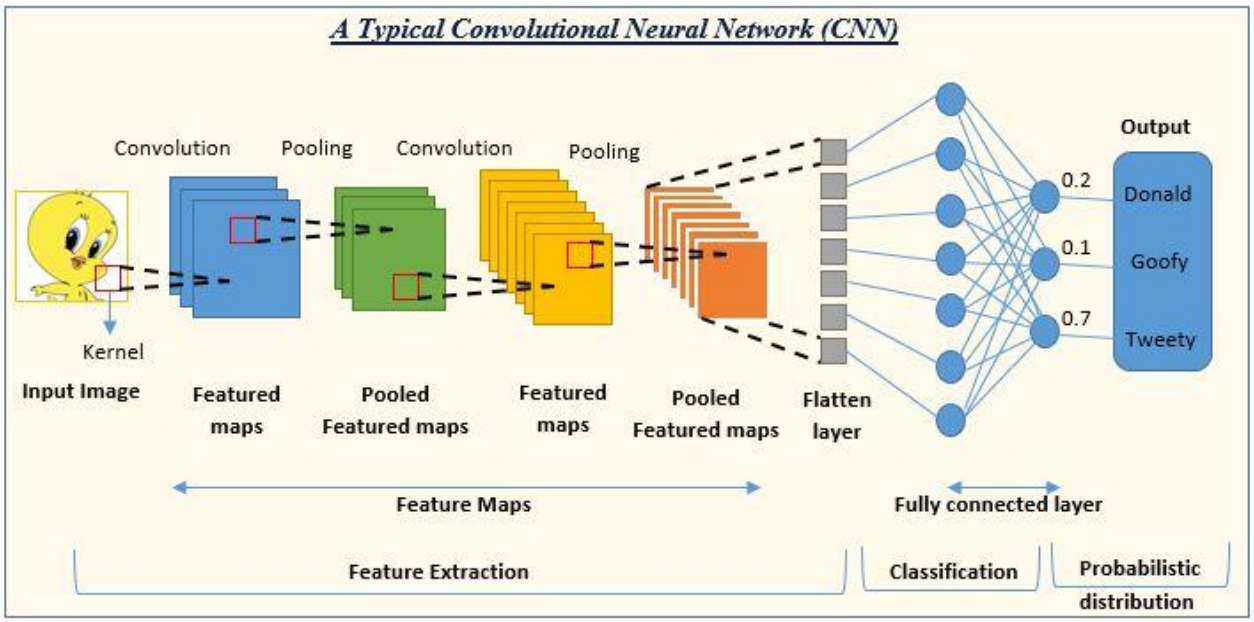
4. Düzleştirme (Smoothing): Derin konvolüsyon ve havuzlama katmanları uygulandıktan sonra, elde edilen özellik haritaları tek boyutlu bir vektör halinde düzleştirilir. Bu, verileri geleneksel bir ileri beslemeli sinir ağına giriş için hazırlar.

5. Tam Bağlantılı Katmanlar: Düzleştirilmiş özellikler daha sonra geleneksel bir sinir ağı gibi hareket eden bir veya daha fazla tam bağlı katmandan geçirilir. Bu katmanlar, giriş görüntüsünü istenen sınıflara (örneğin kedi, köpek, araba) sınıflandırmak için özellikleri birleştirir.

6. Çıktı: Tam bağlı katmanların çıktısı, ham puanları sınıf olasılıklarına dönüştürmek için genellikle bir softmax işlevinden geçirilir. En yüksek olasılığa sahip sınıf daha sonra nihai tahmin olarak seçilir.

7. Eğitim: CNN'ler tipik olarak geri yayılım (Backpropogation) algoritması kullanılarak eğitilir; burada model, etiketli bir veri kümesi üzerinde bir kayıp fonksiyonunu (örneğin çapraz entropi kaybı) en aza indirmek için evrişimli katmanlardaki filtrelerin ağırlıklarını ayarlamayı öğrenir. Bu genellikle stokastik gradyan inişinin (SGD) varyantları kullanılarak yapılır.

CNN'lerin arkasındaki temel fikir, ilk katmanlardaki basit özelliklerden (örneğin kenarlar, renkler) başlayarak daha derin katmanlardaki daha karmaşık özelliklere (örneğin şekiller, nesneler) kadar verilerdeki hiyerarşik örüntüleri otomatik olarak öğrenmektir. Bu da onları görsel bilginin anlaşılması ve işlenmesini gerektiren görevler için oldukça etkili kılmaktadır.



4. Metot ve Uygulama

Bu kısımda sınıflandırma için yapılan uygulamanın metodu anlatılacaktır. Python ve TensorFlow kullanarak modelin oluşturulmasını, eğitilmesini ve sonuçlarını analizini içeren kodlar ve yöntem anlatılacaktır.

- Çalışma Ortamın Ayarlanması: Bu adımda çalışma ortamı için gerekli kütüphanelerin ve bağımlılıkların yüklenmesini ve GPU yapılandırılmasını içerir. Bu kod parçası derin öğrenme için Keras ile TensorFlow'u kullanmak için ortamı ayarlar. `'tf.test.gpu_device_name()'` fonksiyonu TensorFlow'un kullanabileceği bir GPU olup olmadığını kontrol etmek için kullanılmıştır.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
from tensorflow.keras import models, layers

import cv2
import numpy as np
from matplotlib import pyplot as plt

tf.test.gpu_device_name()
```

- Veri Hazırlama: Veri hazırlama kısmı, modelin eğitilmesi için veri kümesinin seçilmesini ve hazırlanmasını içerir. Bu çalışmada, sac metal kusur tespiti için NEU (Northeastern University) veri kümesi seçilmiştir.
- Veri Ön İşleme: Veri kümesi yüklendikten sonra eğitim, doğrulama ve test kümelerine ayrıldı. Eğitim seti, parametrelerini girdi-çıktı çiftlerine göre ayarlayarak modeli eğitmek için kullanılır. Doğrulama seti, modelin parametreleri ayarlamak ve eğitim sırasında performansı değerlendirmek için kullanılır. Test seti, eğitilen modelin görünmeyen veriler üzerindeki nihai performansını değerlendirmek için kullanılır. Bu alt kümeler modelin etkili bir şekilde eğitilmesini, parametrelerin optimize edilmesini ve modelin yeni verilere genelleştirilmesinin doğru bir şekilde değerlendirilmesini sağlar.

```
data_dir = r'data'
IMAGE_SIZE = 200
BATCH_SIZE = 32
CHANNELS = 3

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    directory=data_dir,
    image_size= (IMAGE_SIZE, IMAGE_SIZE),
    batch_size= BATCH_SIZE
)

class_names = dataset.class_names
```

- Veri Artırımı: Veri artırımı (data augmentation), döndürme, çevirme veya ölçekleme gibi dönüşümler yoluyla görüntülerin değiştirilmiş versiyonlarını oluşturarak bir veri kümesinin boyutunu yapay olarak genişletmek için kullanılan ve makine öğrenimi modellerinin sağlamlığını ve genellemesini artırabilen bir tekniktir. Bu çalışmada, görüntüler yatay ve dikey olarak sırasıyla %20'ye kadar (0,2 * 360 derece) çevrilmiş ve döndürülmüş, böylece verilerin çeşitliliği artırılmıştır.

```
data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),])
```

- Model Oluşturma: Bu çalışmada, TensorFlow ve Keras kütüphaneleri kullanılarak bir CNN mimarisi oluşturulmuştur. Model, özellik çıkarma için maksimum havuzlama katmanlarının ardından birkaç konvolüsyonel katmandan ve ardından sınıflandırma için tam bağlı katmanlardan oluşuyordu. Modelin tam iç apısı aşağıdaki gibidir:

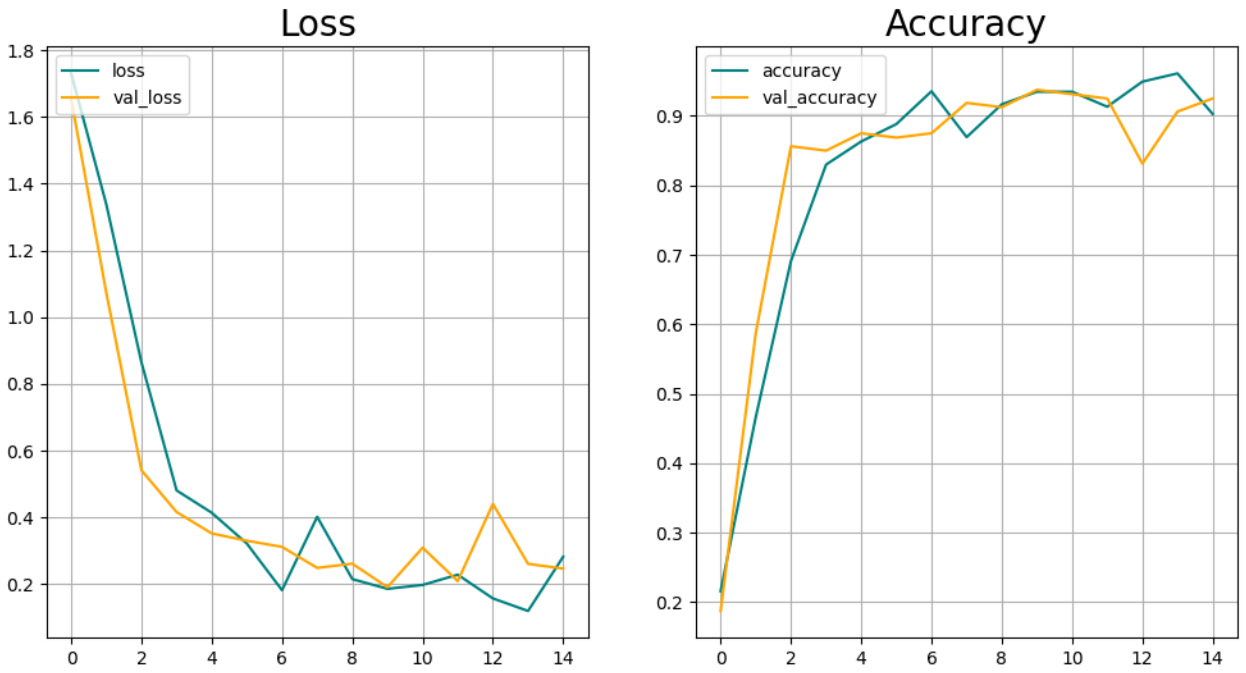
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (32, 200, 200, 3)         0

conv2d (Conv2D)              (32, 198, 198, 32)        896

max_pooling2d (MaxPooling2D  (32, 99, 99, 32)          0)

conv2d_1 (Conv2D)            (32, 97, 97, 64)          18496

max_pooling2d_1 (MaxPooling2D  (32, 48, 48, 64)        0)

conv2d_2 (Conv2D)            (32, 46, 46, 64)          36928

max_pooling2d_2 (MaxPooling2D  (32, 23, 23, 64)        0)

conv2d_3 (Conv2D)            (32, 21, 21, 64)          36928

max_pooling2d_3 (MaxPooling2D (32, 10, 10, 64)         0)

conv2d_4 (Conv2D)            (32, 8, 8, 64)            36928

max_pooling2d_4 (MaxPooling2D  (32, 4, 4, 64)          0)

conv2d_5 (Conv2D)            (32, 2, 2, 64)            36928

max_pooling2d_5 (MaxPooling2D  (32, 1, 1, 64)          0)

flatten (Flatten)            (32, 64)                  0

dense (Dense)                (32, 64)                  4160

dense_1 (Dense)              (32, 6)                   390
=================================================================
Total params: 171,654
Trainable params: 171,654
Non-trainable params: 0
```

- Model Eğitme: Model sparse categorical crossentropy kayıp fonksiyonu ve optimize edici Adam ile derlenmiştir. Eğitim süreci, mini gruplar halinde eğitim seti üzerinde yineleme, kayıp hesaplama ve geriye yayılım kullanarak modelin ağırlıklarını güncellemeyi içeriyordu. Model, aşırı uyumu önlemek için 15 epok boyunca eğitilmiştir. 'model.fit()' fonkisyonu modeli eğitme sürecini başlatır. Burada 15 döngü boyunda, verisetinde tüm verileri gruplar (batch) halinde işleyerek model kendi parametrelerini kaybı minimum tutacak şekilde günceller ve doğrulama (validation) veri grubu ile doğrulama yapar. Model eğitilirken her epok'ta modelin kaybı yavaşça 0'ya yaklaşmakta, aynı zamanda doğruluk oranı da 1'e yaklaşması beklenmektedir.

```
training_history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=15,
    callbacks = [tf.keras.callbacks.TensorBoard(log_dir='logs')]
)
```

5. Sonuçlar ve Doğrulama

Makine öğrenimi modellerinin eğitimi sırasında, epok sayısı arttıkça kayıp azalır ve doğruluk artar. Bunun nedeni, modelin verilerden öğrenmesi ve daha iyi tahminler yapmak için parametrelerini ayarlamasıdır. Oluşturulan modelin eğitimi süresince her epok'taki kaybın ve doğruluğun değerleri aşağıda grafik üzerinde gösterilmiştir. Şekilde görüldüğü gibi döngüler arttıkça, yani model verileri öğrenmeye devam ettikçe kayıplar 0'a düşerken, doğruluk oranı 1'e doğru yükselerek en sonda 0,92 civarında durmuştur.

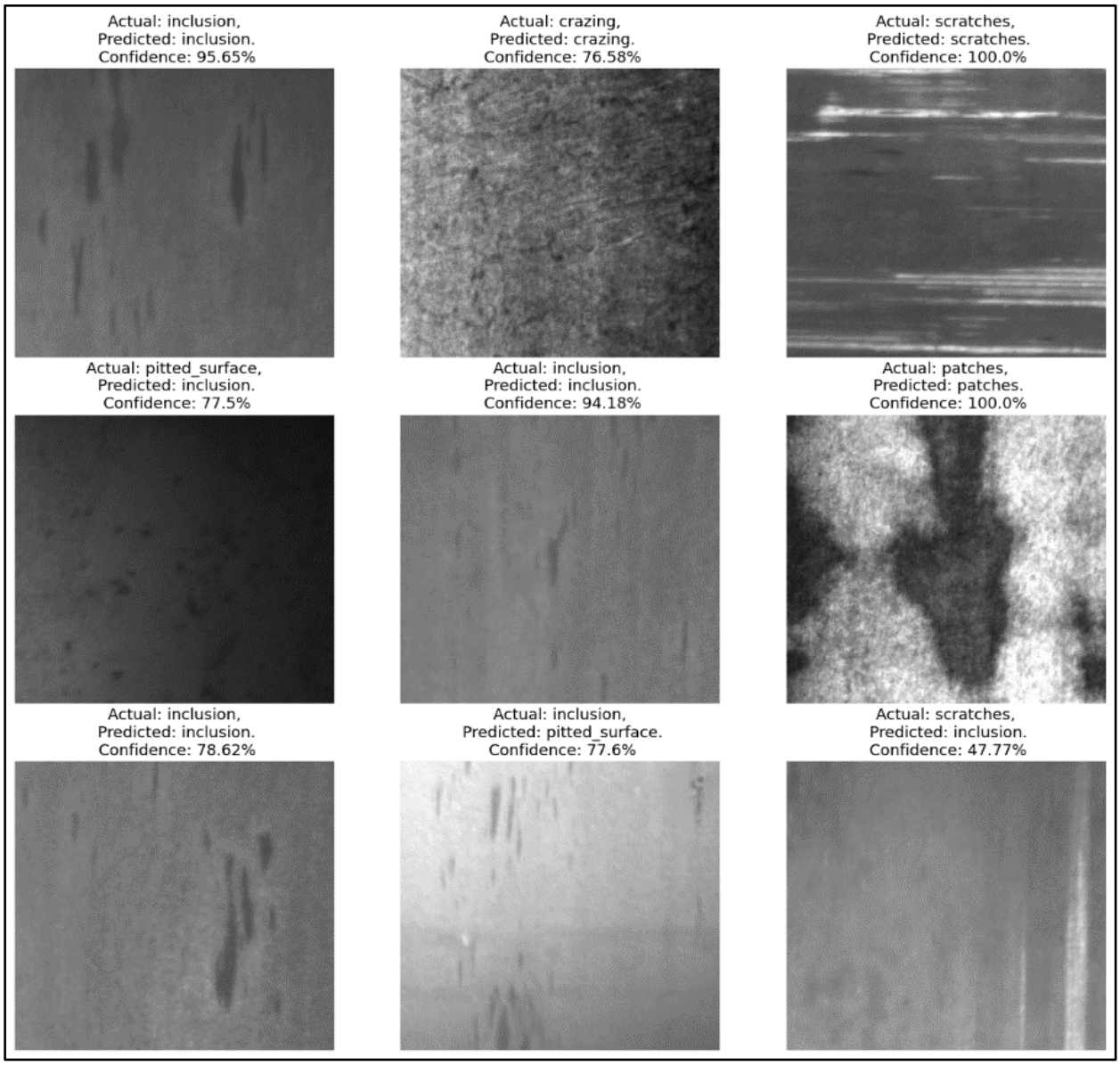


Kayıp ve doğruluk ölçümleri önemlidir çünkü bir makine öğrenimi modelinin ne kadar iyi performans gösterdiğinin nicel bir ölçüsünü sağlarlar. Kayıp metriği, modelin tahminlerinin eğitim verilerindeki gerçek değerlerden ne kadar uzak olduğunu gösterir ve düşük değerler daha iyi performansa işaret eder. Diğer yandan doğruluk metriği, model tarafından yapılan doğru tahminlerin oranını ölçer. Son olarak oluşturulan test veri seti üzerinde modelimizi test ederek modelin doğruluk oranı test edilmiştir.

```
scores = model.evaluate(test_ds)
```

5/5 [==============================] - 1s 18ms/step - loss: 0.2112 - accuracy: 0.9312

Görüldüğü gibi model test veri grubu üzerinde 0,9312 doğrulama oranı ile tahmin yapabilmektedir. Bu tahmin doğruluğunu arttırmak için farklı bir model mimarisi denenebilir veya daha fazla veri toplanarak model daha uzun döngüler boyunca eğitilebilir. Aşağıda test verisindeki fotoğraflar üzerinde model tahmini yapılarak fotoğrafın kendisi gerçek sonuç ve tahmin ile gösterilmiştir.



Referanslar

- Automatic detection and classification of defective areas on metal parts by using adaptive fusion of faster R-CNN and shape from shading. (2022). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/9956999
- Çekiç, İ., & Çavdar, K. (2022). Detection of sheet metal cracks with convolutional artificial neural networks (ESA). Gazi University Journal of Engineering-Architecture Faculty, 38(1), 153-162. https://doi.org/10.17341/gazimmfd.873479
- Real-time steel surface defect recognition based on CNN. (2021, August 23). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9551414
- Pramoditha, R. (2023, February 22). Overview of a neural network's learning process - Data Science 365 - Medium. Medium. https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa
- Shah, S. (2022, March 15). Convolutional Neural Network: an overview. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/
- Wikipedia contributors. (2024, May 1). Neural network (machine learning). Wikipedia. https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
- Metal Surface Defects Dataset. (2020, July 16). Kaggle. https://www.kaggle.com/datasets/fantacher/neu-metal-surface-defects-data
- GitHub - OmarTRBN/MetalDefectClassification: Detecting metal surface defects using Convolutional Neural Networks (CNNs) in Python with TensorFlow. GitHub. https://github.com/OmarTRBN/MetalDefectClassification/tree/master