

2209/A

ÜNİVERSİTE ÖĞRENCİLERİ ARAŞTIRMA PROJELERİ DESTEK  
PROGRAMI

SONUÇ RAPORU

PROJE BAŞLIĞI: **Çelik Levha Yüzeylerindeki Deformasyonların  
Görüntü İşleme Teknikleri ve Derin Öğrenme Yöntemleri ile  
Tespiti**

PROJE YÜRÜTÜCÜSÜNÜN ADI: **Eyüphan Şahin**

DANIŞMANININ ADI: **Dr.Öğr.Üyesi Gülşah Karaduman**

## GENEL BİLGİLER

<b>PROJENİN KONUSU</b>	Çelik levhalar günlük hayatta ve endüstride kullanılan teknolojik ürünlerin üretiminde vazgeçilmez unsurlardır. Otomotiv, inşaat, makine, havacılık ve beyaz eşya gibi üretim sektörlerinde çelik levhalar önemli yer tutmaktadır. Projemizin konusu ise hammadde olarak kullanılan çelik levha yüzeyindeki bozulmaların otomatik olarak görüntü işleme ve derin öğrenme tabanlı bir algoritma ile tespit etmektir.
<b>PROJE YÜRÜTÜCÜSÜNÜN ADI</b>	Eyüphan Şahin
<b>DANIŞMANIN ADI</b>	Dr.Öğr.Üyesi Gülşah Karaduman
<b>PROJE BAŞLANGIÇ VE BİTİŞ TARİHLERİ</b>	

## 1. GİRİŞ

### 1.1. Konu ve Konunun önemi

Günümüzde çelik levhalar günlük hayatta ve endüstride kullanılan teknolojik ürünlerin üretiminde vazgeçilmez unsurlardır. Otomotiv, inşaat, makine, havacılık ve beyaz eşya gibi üretim sektörlerinde çelik levhalar önemli yer tutmaktadır.

Çelik levhalar üretilirken veya çevresel etmenlerden dolayı bu levhaların yüzeylerinde aşınma ve bozulmalar meydana gelebilmektedir. Yüzeyde oluşan aşınma veya bozulma sadece yüzeysel bir aşınma olabileceği gibi içsel bir arıza sonucunda da oluşabilmektedir.

Bu ürünlerde kullanılan çelik levhalardaki yüzey kusurları, soğuk deformasyon ile şekillendirme işlemlerinde çentik etkisi yaratarak ürün kalitesini ve kullanılan kalıpların ömrünü düşürmekte ve üretim için kullanılan enerji miktarını, hammadde israfını artırarak üreticilerde ekonomik kayıplara yol açmaktadır.

Ürün hammaddesi olarak kullanılan bu çelik levhalarda yaşanan herhangi bir kalite sorunu, hayatın neredeyse her alanında kullanılan bu teknolojik ürünlerin üreticilerinde büyük ekonomik ve itibar kayıplarına yol açacaktır.

Bu nedenle, Çelik levhaların kullandığı endüstrilerde, çelik yüzeyindeki aşınma ve bozulmaları görsel ve derin öğrenme tabanlı yöntemlerle tespit edip sınıflandıran sistemler Çelik levhalardaki aşınmaların tespiti için önemli birer araçlardır.

### 1.2. Literatür Analizi

Görüntü tabanlı arıza tespit yöntemleri temassız ve otomatik tespit sistemlerinde kullanıma uygun olduğundan son yıllarda her alanda olduğu gibi çelik yüzeylerde de bozulma tespitinde tercih edilmektedir.

Literatürde çelik yüzeylerdeki bozulmaların tespiti için bazı çalışmalar bulunmaktadır.

Di ve diğerlerinin[2] çelik yüzey hatalarını sınıflandırmak için oluşturdukları modelde yeni bir yarı denetimli öğrenme yöntemi olarak kullandıkları yöntemde iki bileşen öne çıkmaktadır. Bu yöntemlerden birinin Evrişimli Otomatik Kodlama ağı, diğerinin ise yarı denetimli Üretken Düşman Ağı olduğu görülmektedir. Bu yöntemde, Evrişimli Otomatik Kodlayıcı, çok büyük etiketlenmemiş verilerle eğitilmiştir.

Liu et al. [3] çelik yüzeylerin kusurlarını tespit etmek için yeni bir yöntem olarak Haar-Weibull-Variance (HWV) modelini kullanmışlardır. Kusur olmayan ancak görüntüde kusur olarak algılanan sahte kusurları gidermek için bir difüzyon tekniği kullanmışlardır. Daha sonra, yeni bir HWV modeli oluşturarak görüntüdeki her yerel yamanın doku dağılımını karakterize etmeye çalışmışlardır. Çalışmalarında yaptıkları modellerde, sadece iki parametreden başka bir şey kullanmaya gerek kalmadan her bir yamanın doku dağılımını düşük boyutlu uzaya yansıtabilmişlerdir.

He ve ark. [4], hata tespitini gerçekleştirmek için oluşturdukları modelin doğruluğunu artırmak için bir sınıflandırma ağına yanı sıra bir sınıflandırma öncelikli ağ ve birçok grulu CNN'nin yanı sıra bir nesne algılama çerçeve yapısı gerçekleştirmiştir. Görüntüyü, öncelikli ağdaki farklı türdeki kusurları eşleştirmek için farklı evrişim çekirdek gruplarını ayrı ayrı eğiten çok grulu CNN ile sınıflandırmışlardır. Elde ettikleri sınıflandırma sonuçları doğrultusunda, hata sınıfının etiketini yeniden düzenlemek için YOLO tabanlı bir sinir ağına girdi olarak hata içerdiğini düşündükleri öznelik haritalarını vermişlerdir.

Dong ve diğerleri[5], çelik şeritler üzerindeki yüzey kusurlarını piksel bazında ve genel yapıyı tespit etmek için PGA-net adıyla oluşturdukları modelde küresel bağlam dikkat ağı ile birlikte piramit öznelik füzyonunu kullanmışlardır. Geliştirdikleri ağda çok ölçekli özellikleri çıkarmaktadırlar. Bu özellikleri yoğun atlama bağlantıları aracılığıyla birleştirmek için piramit özellikli bir füzyon modülü kullanmaktadırlar.

Wang ve ark. [6] çelik şerit yüzeylerinin tespiti için bir şablon tekniği önermektedirler. Bu şablon tekniğine dayalı basit bir algoritma geliştirmişlerdir. Çalışmalarında, mükemmel dokuların istatistiksel özelliklerini çıkarmak için önce çok sayıda hatasız görüntü toplamışlardır. Ardından, test görüntülerinin istatistiksel özelliklerine ve boyutlarına göre her bir test görüntüsü için şablonlar oluşturmaktadırlar. Luo ve ark. [7], genelleştirilmiş tamamen yerel ikili modeller olarak adlandırılan bir görüntüdeki kusurları tespit eden bir çerçeve tasarlamışlardır. Çelik yüzeylerdeki kusurları sınıflandırmak için bu çerçevede iki alt model geliştirmişlerdir. Bu modellerden biri gürültü ile değişmeyen yerel yapı modelidir.

He ve diğerleri[8] çelik plakalarda sadece yüzey bozulmalarını tespit etmek için CNN ve çok seviyeli özellik füzyon ağı yöntemlerini birleştirerek bir uygulama gerçekleştirmektedirler. Çalışmalarında, başarılı bir sınıflandırma yapmak ve her aşamada öznelik tablosu oluşturmak için temel CNN'ni kullanmakta ve ardından MFN çoklu hiyerarşik özelliklerini, kusurların daha fazla konum detayını içerebilecek tek bir özellikte birleştirmektedirler. Bu çok seviyeli özelliklerden yararlanarak, ilgi alanları oluşturmak için bölgesel bir teklif ağı oluşturmışlardır.

Badrinarayanan ve ark. [9] piksel tabanlı semantik segmentasyon için SegNet adlı pratik, derin tam bir CNN modeli tasarlamışlardır. Zhao ve ark. [10], karmaşık sahne yapısından ek bağlamsal bilgileri çıkarmak için bir piramit sahne ayrıştırma ağı (PSPNet) modeli ve bir küresel piramit havuzu özelliği kullanmışlardır. Ayrıca ResNet tabanlı tam evrişimli ağ (FCN) için derin kontrollü bir optimizasyon gerçekleştirmişlerdir.

Lin ve ark. [13], uzun menzilli yedek bağlantılar kullanarak yüksek çözünürlüklü tahmin sağlamak için alt örnekleme sırasında mevcut tüm bilgileri kullanan genel birçok yollu iyileştirme ağı olarak tanımlanan Re-fineNet modelini tasarlamışlardır. Fu ve ark. [14], düşük seviyeli özelliklerin eğitimini vurgulayarak çelik yüzey kusurlarını sınıflandırmak için bir CNN modeli ve çoklu alıcı alanlar oluşturmışlardır. Önerdikleri yöntemde omurga mimarisi olarak CNN yapısına sahip SqueezeNet kullanmışlardır. Bu ağ önceden eğitilmiş bir ağıdır. Kamera gürültüsü, farklı aydınlatma ortamları ve hareket bulanıklığı gibi çelik yüzey kusurlarını artırarak test verilerini artırmışlardır.

Song ve ark. [15], metal bileşenlerdeki zayıf çizgileri erken tespit etmek ve büyük sorunları önlemek için derin CNN ile bir algılama modeli gerçekleştirmiştir. Zhou ve ark. [16], karmaşık doku problemini, değişken girişim faktörlerini ve ayrıca büyük örnekleme probleminin çözümünü çözmeyi amaçlayan ikili bir modele dayalı olarak yüzey kusurlarını otomatik olarak tespit edebilen genel bir yöntem geliştirmişlerdir.

### 1.3. Özgün Değer

Bu projede çelik levha yüzeylerinde oluşan/oluşabilecek aşınma ve bozulmaların görüntüden tespiti için görüntü işleme ve derin öğrenme tabanlı bir yöntem geliştirilmiştir. Çelik levha yüzey görüntülerinden oluşan ve 6 tip arıza sınıfı bulunan bir veri seti kullanılarak bir CNN mimarisi eğitilip ve test edilmiştir.

Bu derin öğrenme mimarisinin eğitim ve test işlemleri ve görüntü işleme teknikleri daha hızlı yapabilmek adına JetsonNano kitinde gerçekleştirilip eğitilen ve test edilen modeli gerçek zamana uyarlayabilmek için yapay zekâ kitine bağlı bir kamera ile çelik yüzey görüntüsü alınmış ve bu görüntüdeki arızanın tanınması ve sınıflandırılması sağlanmıştır.

### 1.4. Araştırma sorumuz:

Derin öğrenme teknolojisini kullanarak çelik levhalardaki 6 adet kusur sınıfı olan; hadde çizizi, yüzeysel pürüzler, yüzeyde oluşan oyuklar, içirme ve çentiklerin tespit edilmesi.

## 2. Rapor dönemlerinde yapılan çalışmalar

Projeye başlarken dikkat ettiğimiz adımlar aşağıda sıralanmıştır.

**Yapay Zekâ Geliştirme Kitinin kurulumu veri setinin elde edilmesi**

**Derin öğrenme tabanlı sınıflandırma algoritmalarının geliştirilmesi**

**Algoritmaların test edilmesi**

**Sonuçların birleştirilmesi ve projenin raporlanması**

### Yapay Zekâ Geliştirme Kitinin kurulumu veri setinin elde edilmesi

Çalışmada, çelik yüzeylerdeki aşınma ve bozulmaların tespiti için Jetson Nano kullanan gömülü bir tespit sistemi geliştirilmiştir. Derin öğrenme uygulamalarının artmasıyla birlikte derin öğrenme algoritmaları, Jetson Nano gibi gömülü platformlarda aktif olarak çalıştırılmaktadır. Jetson Nano, yüksek hızlı GPU'lara sahip olduğu için yüksek performansa sahiptir. Jetson kartları, hafiflik ve taşınabilirlik, enerji verimliliği ve güç tüketimi başına yüksek performans gibi avantajlara sahiptir. Sistemde, kamera olarak Raspberry Pi Kamera Modülü V2 kullanılmaktadır. Jetson Nano ve kamera konfigürasyonu, gelişmiş görüntü kalitesi ile çok sayıda uygulama için kullanılabilecek ham veri formatında görüntüler sağlayabildiğinden, kusur tespiti için performansının değerlendirilmesi uygun görülmüştür. Kullanılan kamera 3280 x 2464 piksel çözünürlükte statik görüntüler üretebilmektedir. Video kaydında 1080p 30fps, 720p 60fps ve 640x480p 60fps'yi destekler. Kameradaki IMX219 sensörü, 400 ile 700 nm arasındaki görünür spektral aralıkta çalışmaktadır.V2 kamera modülü 25mm×25mm×9mm boyutlarında ve yaklaşık 3g ağırlığındadır. Kusur tespit sisteminin donanım bileşimi aşağıdaki şekilde gösterilmektedir.



Şekil 1. Jetson Nano ve Raspberry Pi kamera modülü V2

Toplanan Derin öğrenme tabanlı sınıflandırma algoritmalarının geliştirilmesi

## Import Dependencies

```
# Import dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import datetime
import itertools
import random
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
```

Projede kullanılacak kütüphanelerin projeye eklenmesi.

```
!unzip NEU.zip

Archive:  NEU.zip
  creating:  NEU/
  inflating: NEU/Thumbs.db
  inflating: __MACOSX/NEU/._Thumbs.db
  creating:  NEU/valid/
  inflating: __MACOSX/NEU/._valid
  inflating: NEU/.DS_Store
  inflating: __MACOSX/NEU/._.DS_Store
  creating:  NEU/test/
  inflating: __MACOSX/NEU/._test
  creating:  NEU/train/
  inflating: __MACOSX/NEU/._train
  creating:  NEU/valid/Inclusion/
  inflating: __MACOSX/NEU/valid/._Inclusion
  creating:  NEU/valid/Patches/
  inflating: __MACOSX/NEU/valid/._Patches
  creating:  NEU/valid/Rolled/
  inflating: __MACOSX/NEU/valid/._Rolled
  creating:  NEU/valid/Pitted/
  inflating: __MACOSX/NEU/valid/._Pitted
  creating:  NEU/valid/Scratches/
  inflating: __MACOSX/NEU/valid/._Scratches
  creating:  NEU/valid/Crazing/
  inflating: __MACOSX/NEU/valid/._Crazing
  inflating: NEU/test/.DS_Store
  ...
  inflating: NEU/train/Crazing/Cr_221.bmp
  inflating: __MACOSX/NEU/train/Crazing/._Cr_221.bmp
  inflating: NEU/train/Crazing/Cr_61.bmp
  inflating: __MACOSX/NEU/train/Crazing/._Cr_61.bmp
```

(NEU.zip) dosyasını açıp içeriğini aktarır İçerik “NEU” adında bir dizin ve alt dizinleri içeriyor

- "NEU" dizini, veri setini içerir.
- "valid" dizini, doğrulama veri setini içerir.
- "test" dizini, test veri setini içerir
- "train" dizini, eğitim veri setini içerir.

## Load and Walk Through the data

```
for dirpath, dirnames, filenames in os.walk('/content/drive/MyDrive/TUBITAK EYUP/NEU/'):
    print(f"There are {len(dirnames)} directories and {len(filenames)} images in '{dirpath}'.")

There are 3 directories and 2 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/'.
There are 6 directories and 0 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Inclusion'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Patches'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Rolled'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Pitted'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Scratches'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Crazing'.
There are 6 directories and 1 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Inclusion'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Patches'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Rolled'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Pitted'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Scratches'.
There are 0 directories and 12 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Crazing'.
There are 6 directories and 1 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Inclusion'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Patches'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Rolled'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Pitted'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Scratches'.
There are 0 directories and 276 images in '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Crazing'.
```

Bu kod bloğu, /content/drive/MyDrive/TUBITAK EYUP/NEU/ dizinindeki dosya ve dizin yapısını gezinerek her bir dizin için içerdikleri dosya ve dizin sayısını yazdırır.

```
# Check how many images are there in a particular class (can be done for other classes!)
num_of_inclusion_images_train = len(os.listdir("/content/drive/MyDrive/TUBITAK EYUP/NEU/train/Inclusion"))
num_of_inclusion_images_test = len(os.listdir("/content/drive/MyDrive/TUBITAK EYUP/NEU/test/Inclusion"))
num_of_inclusion_images_val = len(os.listdir("/content/drive/MyDrive/TUBITAK EYUP/NEU/valid/Inclusion"))
print(f"Total number of test images of inclusion is {num_of_inclusion_images_test} \nTotal number of train images of inclusion is {num_of_inclusion_images_train} \nTotal number of v

Total number of test images of inclusion is 12
Total number of train images of inclusion is 276
Total number of validation images of inclusion is 12
```

Bu kod bloğu, belirli bir sınıfa ait olan görüntü dosyalarının eğitim, test ve doğrulama veri setlerindeki sayısını göstermektedir.

os.listdir() fonksiyonu, belirtilen dizindeki dosya ve dizinleri listeler. Bu durumda, belirli bir sınıfın eğitim, test ve doğrulama dizinlerindeki görüntü dosyalarının sayısını almak için kullanılır.

len() fonksiyonu, bir liste veya dize gibi bir nesnenin uzunluğunu döndürür. Bu durumda, os.listdir() ile alınan dosya listelerinin uzunluğu, yani görüntü dosyalarının sayısı hesaplanır.

Sonuçlar print() ifadesiyle yazdırılır ve ilgili sınıfa ait olan eğitim, test ve doğrulama veri setlerindeki görüntü sayıları görüntülenir.

## Visualize A Random Image from The Dataset

```
# Define a function to view a random image
def viewRandomImage(targetDir, targetClass):
    # Setup target directory
    targetFolder = targetDir + targetClass

    # Get a random image path
    randomImage = random.sample(os.listdir(targetFolder), 1)

    # Read in the image and plot it using matplotlib
    img = mpimg.imread(targetFolder + '/' + randomImage[0])
    plt.imshow(img)
    plt.title(targetClass)
    plt.axis(False)

    # Show the shape of the image
    print(f"Image Shape: {img.shape}")

    return img
```

Bu kod bloğu, rastgele bir görüntüyü görüntülemek için bir fonksiyon tanımlar. Seçilen rastgele görüntüyü okur ve mpimg.imread() fonksiyonunu kullanarak matplotlib ile görüntüler. Görüntünün başlığını hedef sınıf olarak ayarlar, eksenleri kapatarak görüntünün eksen çizgilerini gizler. Son olarak, görüntünün şeklini (img.shape) yazdırır ve görüntüyü döndürür.

```
# View a random image from the training dataset (can be used for other classes!)
img = viewRandomImage(targetDir = '/content/drive/MyDrive/TUBITAK EYUP/NEU/train/',
                        targetClass = 'Inclusion')
```

Bu kod bloğu, eğitim veri setinden (/content/drive/MyDrive/TUBITAK EYUP/NEU/train/) rastgele bir görüntüyü göstermek için önceki tanımlanan viewRandomImage() fonksiyonunu kullanır.

## Data Preparation

```
# Setup the training and the test directory paths
trainDir = "/content/drive/MyDrive/TUBITAK EYUP/NEU/train"
valDir = "/content/drive/MyDrive/TUBITAK EYUP/NEU/valid"
testDir = "/content/drive/MyDrive/TUBITAK EYUP/NEU/test"

# Define the hyperparameters
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

# Create the train and test data
train_data = tf.keras.preprocessing.image_dataset_from_directory(directory = trainDir,
                                                                    image_size = IMG_SIZE,
                                                                    label_mode = 'categorical',
                                                                    batch_size = BATCH_SIZE)

val_data = tf.keras.preprocessing.image_dataset_from_directory(directory = valDir,
                                                                image_size = IMG_SIZE,
                                                                label_mode = 'categorical',
                                                                batch_size = BATCH_SIZE,
                                                                shuffle = False)

test_data = tf.keras.preprocessing.image_dataset_from_directory(directory = testDir,
                                                                image_size = IMG_SIZE,
                                                                label_mode = 'categorical',
                                                                batch_size = BATCH_SIZE,
                                                                shuffle = False)
```

Bu kod bloğu, eğitim ve test dizinlerinin yolunu ayarlar ve bu dizinlerden TensorFlow için veri kümesi oluşturur. trainDir, valDir ve testDir değişkenleri, eğitim, doğrulama ve test dizinlerinin tam yolunu içerir. IMG\_SIZE değişkeni, görüntülerin yeniden boyutlandırılacağı hedef boyutu (224x224) temsil eder. BATCH\_SIZE değişkeni, veri kümesinin her bir yığınının kaç görüntünün bulunacağını belirler. tf.keras.preprocessing.image\_dataset\_from\_directory() fonksiyonu, veri dizinlerinden TensorFlow için veri kümesi oluşturmak için kullanılır. Bu fonksiyon, görüntüleri otomatik olarak yükleme, yeniden boyutlandırma, sınıf etiketlerini kategorik hale getirme ve veriyi yığınlar halinde alma işlemlerini yapar. train\_data, val\_data ve test\_data değişkenleri, sırasıyla eğitim, doğrulama ve test veri kümelerini temsil eder.



## Creating the Callbacks

```
# Create the Tensorboard callback
def create_tensorboard_callback(dir_name, experiment_name):
    log_dir = dir_name + "/" + experiment_name + "/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    tensorboard_callback = tf.keras.callbacks.TensorBoard(
        log_dir=log_dir)
    print(f"Saving TensorBoard log files to: {log_dir}")
    return tensorboard_callback
```

```
# Create the checkpoint path
checkpoint_path = 'surface_defects_detection_model_checkpoint/checkpoint.ckpt'
checkpoint_callback = ModelCheckpoint(checkpoint_path,
                                     save_weights_only=True,
                                     monitor='val_accuracy',
                                     save_best_only=True)
```

```
class CutTraining(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_accuracy') > 0.80):
            print('\nReached more than 80% accuracy so cancelling training!')
            surface_defects_detection_model.stop_training = True

cut_training = CutTraining()
```

**İlk kod bloğu:** Bu kod bloğu Training loglarını özel seçilmiş bir directory ismi tanımlayarak oraya kaydeder.

**İkinci kod bloğu:** modelin kontrol noktası yolunu oluşturmak için bir değişken tanımlar ve ModelCheckpoint geri çağırısını oluşturur.

**Üçüncü kod bloğu:** Bu kod bloğu, validation Accuracy %80 i geçtiği zaman trainingi kesmek için tanımlanmıştır

## Creating the Model

```
# Create the model
surface_defects_detection_model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (2,2), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (2,2), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (2,2), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    #tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(6, activation='softmax')
])

# Compile the model
surface_defects_detection_model.compile(loss = 'categorical_crossentropy',
                                       optimizer = 'Adam',
                                       metrics = ['accuracy'])

# Fit the model
history_surface_defects_detection_model = surface_defects_detection_model.fit(train_data,
                                       epochs = 50,
                                       steps_per_epoch = len(train_data),
                                       validation_data = val_data,
                                       shuffle=True,
                                       # validation_steps = len(val_data),
                                       callbacks = [create_tensorboard_callback(dir_name = 'training_logs', experiment_name = 'surface_defects_detection_model'), cut_training])
```

```

Saving TensorBoard log files to: training_logs/surface_defects_detection_model/20230529-152206
Epoch 1/50
6/52 [==>.....] - ETA: 3s - loss: 479.0736 - accuracy: 0.1458
WARNING:tensorflow:Callback method 'on_train_batch_end' is slow compared to the batch time (batch time: 0.0279s vs 'on_train_batch_end' time: 0.0381s). Check your callbacks.
52/52 [=====] - 10s 90ms/step - loss: 68.9583 - accuracy: 0.2156 - val_loss: 2.0085 - val_accuracy: 0.2361
Epoch 2/50
52/52 [=====] - 4s 66ms/step - loss: 1.4790 - accuracy: 0.4203 - val_loss: 1.1801 - val_accuracy: 0.5417
Epoch 3/50
52/52 [=====] - 4s 70ms/step - loss: 1.1601 - accuracy: 0.5537 - val_loss: 1.0960 - val_accuracy: 0.4583
Epoch 4/50
52/52 [=====] - 5s 78ms/step - loss: 0.9487 - accuracy: 0.6165 - val_loss: 0.9674 - val_accuracy: 0.5833
Epoch 5/50
52/52 [=====] - 4s 66ms/step - loss: 0.7914 - accuracy: 0.6612 - val_loss: 0.7086 - val_accuracy: 0.6667
Epoch 6/50
52/52 [=====] - 4s 73ms/step - loss: 0.6494 - accuracy: 0.7428 - val_loss: 0.7375 - val_accuracy: 0.6944
Epoch 7/50
52/52 [=====] - 5s 77ms/step - loss: 0.7965 - accuracy: 0.6963 - val_loss: 1.2194 - val_accuracy: 0.5000
Epoch 8/50
52/52 [=====] - 4s 67ms/step - loss: 0.6341 - accuracy: 0.7234 - val_loss: 0.6169 - val_accuracy: 0.7500
Epoch 9/50
52/52 [=====] - 4s 66ms/step - loss: 0.4116 - accuracy: 0.8400 - val_loss: 0.6426 - val_accuracy: 0.7361
Epoch 10/50
52/52 [=====] - 5s 74ms/step - loss: 0.4473 - accuracy: 0.8207 - val_loss: 0.9336 - val_accuracy: 0.6667
Epoch 11/50
52/52 [=====] - 4s 68ms/step - loss: 0.2871 - accuracy: 0.8835 - val_loss: 1.0066 - val_accuracy: 0.6111
Epoch 12/50
52/52 [=====] - 4s 73ms/step - loss: 0.2256 - accuracy: 0.8967 - val_loss: 0.8473 - val_accuracy: 0.7222
Epoch 13/50
52/52 [=====] - 5s 75ms/step - loss: 0.2145 - accuracy: 0.9106 - val_loss: 0.7189 - val_accuracy: 0.7222
...
Epoch 20/50
52/52 [=====] - ETA: 0s - loss: 0.1896 - accuracy: 0.9088
Reached more than 80% accuracy so cancelling training!
52/52 [=====] - 4s 69ms/step - loss: 0.1896 - accuracy: 0.9088 - val_loss: 0.7735 - val_accuracy: 0.8194
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```

Bu kod bloğu, yüzey kusurları tespit modelinin tanımlandığı, derlendiği ve eğitildiği bölümleri içerir.

Model, `tf.keras.models.Sequential` sınıfını kullanarak oluşturulur. Ardışık katmanlar sırasıyla eklenir. Modelin mimarisi aşağıdaki gibi tanımlanır:

İlk katman, 32 filtre ve 2x2 boyutunda bir Conv2D katmanıdır. ReLU aktivasyonu kullanır ve giriş boyutu (224, 224, 3) olarak belirlenir.

İkinci katman, 2x2 boyutunda bir MaxPooling2D katmanıdır.

Üçüncü katman, 64 filtre ve 2x2 boyutunda bir Conv2D katmanıdır. ReLU aktivasyonu kullanır.

Dördüncü katman, 2x2 boyutunda bir MaxPooling2D katmanıdır.

Beşinci katman, 128 filtre ve 2x2 boyutunda bir Conv2D katmanıdır. ReLU aktivasyonu kullanır.

Altıncı katman, 2x2 boyutunda bir MaxPooling2D katmanıdır.

Yedinci katman, Flatten katmanıdır ve verileri düzleştirir.

Sekizinci katman, 256 nöronlu bir Dense (tam bağlı) katmandır ve ReLU aktivasyonu kullanır.

Dokuzuncu katman, 128 nöronlu bir Dense katmandır ve ReLU aktivasyonu kullanır.

Onuncu katman, 64 nöronlu bir Dense katmandır ve ReLU aktivasyonu kullanır.

On birinci katman, 6 nöronlu bir Dense katmandır ve softmax aktivasyonu kullanır. Çıkış sınıflarının sayısı olan 6'ya karşılık gelir.

Model derlenirken, kayıp fonksiyonu olarak "categorical\_crossentropy", optimizier olarak "Adam" ve doğruluk metriği olarak "accuracy" belirtilir.

Model eğitimi `surface_defects_detection_model.fit` yöntemi kullanılarak gerçekleştirilir. Eğitim verileri `train_data` kullanılır ve 52 epoch boyunca eğitim yapılır. Eğitim sırasında her epoch sonunda doğrulama verileri `val_data` kullanılarak doğruluk değeri hesaplanır. Eğitim süreci TensorBoard geri çağırısı ve `cut_training` geri çağırısıyla takip edilir.

## Model Summary

```
surface_defects_detection_model.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 223, 223, 32)	416
max_pooling2d_3 (MaxPooling 2D)	(None, 111, 111, 32)	0
conv2d_4 (Conv2D)	(None, 110, 110, 64)	8256
max_pooling2d_4 (MaxPooling 2D)	(None, 55, 55, 64)	0
conv2d_5 (Conv2D)	(None, 54, 54, 128)	32896
max_pooling2d_5 (MaxPooling 2D)	(None, 27, 27, 128)	0
flatten_1 (Flatten)	(None, 93312)	0
dense_4 (Dense)	(None, 256)	23888128
dense_5 (Dense)	(None, 128)	32896
...		
Total params:	23,971,238	
Trainable params:	23,971,238	
Non-trainable params:	0	

Kod, oluşturulan modelin özetini ekrana yazdırır. Bu özet, modelin katmanlarını, her katmandaki çıktı şekillerini, katmanlardaki parametre sayılarını ve toplam öğrenilebilir parametre sayısını içerir. Ayrıca, her katmanın isimlerini ve aktivasyon fonksiyonlarını da gösterir.

## Model Evaluation

```
# Evaluate our model on the whole test data
results_surface_defects_detection_model = surface_defects_detection_model.evaluate(val_data, verbose = 0)
print("Validation Loss: {:.3f}, Validation Accuracy: {:.3f}".format(results_surface_defects_detection_model[0], results_surface_defects_detection_model[1]))
```

```
Validation Loss: : 0.774, Validation Accuracy: 0.819
```

Kod, eğitilmiş `surface_defects_detection_model` modelinin doğrulama verisi (`val_data`) üzerinde değerlendirilmesini ve doğrulama kaybını ve doğruluk değerini ekrana yazdırır. `evaluate` fonksiyonunu kullanarak modelin verilen veri üzerindeki kaybını ve doğruluğunu hesaplar. Sonuçlar daha sonra `format` fonksiyonuyla biçimlendirilir ve `print` fonksiyonuyla ekrana yazdırılır.

# Visualization Loss Curves

```
# Define a function to plot loss curves
def plot_loss_curves(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']

    epochs = range(len(history.history['loss']))

    # Plot loss
    plt.figure(figsize = (15, 6))
    plt.subplot(1, 2, 1)
    plt.plot(epochs, loss, label='training_loss')
    plt.plot(epochs, val_loss, label='val_loss')
    plt.title('Training Loss vs Validation Loss')
    plt.xlabel('Epochs')
    plt.legend()

    # Plot accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs, accuracy, label='training_accuracy')
    plt.plot(epochs, val_accuracy, label='val_accuracy')
    plt.title('Training Accuracy vs Validation Accuracy')
    plt.xlabel('Epochs')
    plt.legend();
```

22]

Bu kod parçası, eğitim geçmişine dayanarak kayıp (loss) ve doğruluk (Accuracy) eğrilerini çizmek için kullanılan bir işlevi tanımlar. İşlev, eğitim sürecindeki kayıp ve doğrulama kaybı değerlerini alır ve bunları grafik olarak çizer. Ayrıca, eğitim doğruluğu ve doğrulama doğruluğu değerlerini de grafik olarak çizer. Böylece, modelin eğitim sürecindeki performansını izlemek ve eğitim ilerlemesini görselleştirmek için kullanılabilir.

## Algoritmaların test edilmesi

# Making predictions on the test data

```
# Create a function to load and prepare images
def load_and_prep_image(filename, img_shape = 224, scale = True):
    # Read in image
    img = tf.io.read_file(filename)

    # Decode image into tensor
    img = tf.io.decode_image(img, channels = 3)

    # Resize the image
    img = tf.image.resize(img, size = [img_shape, img_shape])

    # Scale? Yes or No
    if scale:
        # rescale the image (get all values between 0 and 1)
        return img/255.
    else:
        return img
```

4]

Bu kod, belirtilen bir dosyadan görüntü yüklemek ve hazırlamak için `load_and_prep_image` adlı bir işlev tanımlar. İşlev, belirtilen dosyanın okunması, çözülmesi, yeniden boyutlandırılması ve gerektiğinde ölçeklendirilmesi gibi işlemleri gerçekleştirir.

İşlev, belirtilen dosyayı okur, görüntüyü bir tensöre dönüştürür, hedef boyuta yeniden boyutlandırır ve gerektiğinde ölçeklendirir. Ölçekleme işlemi, görüntüyü 0 ile 1 arasında değerlere dönüştürerek işleme kolaylığı sağlar.

```
# Get the class names
class_names = test_data.class_names

# Make preds on a series of random images
plt.figure(figsize = (25, 12))
for i in range(5):
    # Choose random image(s) from random class(es)
    class_name = random.choice(class_names)
    filename = random.choice(os.listdir(testDir + '/' + class_name))
    filepath = testDir + '/' + class_name + '/' + filename

    # Load the image and make predictions
    img = load_and_prep_image(filepath, scale = True) # efficientnet
    pred_prob = surface_defects_detection_model.predict(tf.expand_dims(img, axis = 0)) # get prediction probabilities array
    pred_class = class_names[pred_prob.argmax()] # get the highest prediction probability index and match it class_names list

    # Plot the image(s)
    plt.subplot(1, 5, i+1)
    plt.imshow(img/255.)
    if class_name == pred_class: # if predicted class matches truth class, make text green
        title_color = 'g'
    else:
        title_color = 'r'
    plt.title(f'Actual: {class_name}, pred: {pred_class}, prob: {pred_prob.max():.2f}', c = title_color)
    plt.axis(False);

1/1 [=====] - 0s 277ms/step
1/1 [=====] - 0s 33ms/step
1/1 [=====] - 0s 28ms/step
1/1 [=====] - 0s 33ms/step
1/1 [=====] - 0s 33ms/step
```

Bu kod, test veri kümesinden rastgele görüntüler seçerek, bu görüntüler üzerinde tahminler yapmayı ve sonuçları görselleştirmeyi sağlar.

Kodun işleyişi şu şekildedir:

İlk olarak, test veri kümesinin sınıf isimlerini `class_names` değişkenine atarız. Ardından, bir döngü içinde belirli bir sayıda (burada 5) rastgele görüntü seçeriz. Her bir döngü adımında, rastgele bir sınıf adı seçeriz (`class_name`) ve bu sınıfa ait rastgele bir görüntü dosyası adı seçeriz (`filename`). Görüntünün dosya yolunu oluştururuz (`filepath`). `load_and_prep_image` işlevini kullanarak, seçilen görüntüyü yükler ve hazırlarız. Hazırlanan görüntü üzerinde `surface_defects_detection_model` modelini kullanarak tahmin yaparız. Bu, tahmin olasılıkları dizisini (`pred_prob`) elde etmemizi sağlar. En yüksek tahmin olasılığına sahip sınıfı (`pred_class`) `class_names` listesiyle eşleştiririz. Görüntüyü ve tahmin sonuçlarını görselleştiririz. Eğer tahmin edilen sınıf gerçek sınıf ile eşleşiyorsa, metin rengini yeşil yaparız, aksi takdirde kırmızı yaparız.

## Analyzing the classification reports and Plotting the confusion matrix

```
# Make a couple of preparations
yLabels = []
for images, labels in test_data.unbatch():
    yLabels.append(labels.numpy().argmax())

predProbs = surface_defects_detection_model.predict(test_data)
predClasses = predProbs.argmax(axis = 1)

# Check the classification reports by printing it out
print(classification_report(y_true = yLabels, y_pred = predClasses))
```

```
3/3 [=====] - 0s 38ms/step
      precision    recall  f1-score   support

     0       0.79       0.92       0.85        12
     1       0.50       0.33       0.40        12
     2       0.90       0.75       0.82        12
     3       0.50       0.67       0.57        12
     4       0.80       0.67       0.73        12
     5       0.79       0.92       0.85        12

 accuracy          0.71
 macro avg         0.71       0.71       0.70
 weighted avg      0.71       0.71       0.70
```

Bu kod parçası, test veri kümesi üzerindeki tahmin sonuçlarıyla gerçek etiketler arasında sınıflandırma raporunu hesaplar ve yazdırır. Bu parametrelere bakılma sebebi çoğu projede sadece accuracy'e bakmak doğru bir sonuç vermemektedir. Bu kod parçasını kullanarak, modelin performansını daha ayrıntılı bir şekilde değerlendirebiliriz Sınıflandırma raporu, her bir sınıf için hassasiyet (precision), geri çağırma (recall), F1 skoru ve destek (support) gibi performans metriklerini sağlar. Bu rapor, modelin sınıflandırma performansını daha iyi anlamamıza yardımcı olur

```
# Get a dataframe of the classification report
classification_report_df = pd.DataFrame(dict(classification_report(yLabels, predClasses, output_dict = True)))
classification_report_df.T
```

Bu kod, classification\_report işlevinden elde edilen sınıflandırma raporunu bir DataFrame'e dönüştürür. output\_dict=True parametresiyle birlikte kullanarak, sınıflandırma raporunu bir sözlük olarak alır ve bu sözlüğü kullanarak DataFrame oluşturur. Son olarak, .T özelliğini kullanarak DataFrame'in transpozunu alır ve sınıfların satırlar ve metriklerin sütunlar olarak görüntülenmesini sağlar.

```

# Define a function to plot a confusion matrix
def make_confusion_matrix(y_true, y_pred, classes=None, figsize=(10, 10), text_size=15, norm=False, savefig=False):
    # Create the confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    cm_norm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] # normalize it
    n_classes = cm.shape[0] # find the number of classes we're dealing with

    # Plot the figure and make it pretty
    fig, ax = plt.subplots(figsize=figsize)
    cax = ax.matshow(cm, cmap=plt.cm.Blues) # colors will represent how 'correct' a class is, darker == better
    fig.colorbar(cax)

    # Are there a list of classes?
    if classes:
        labels = classes
    else:
        labels = np.arange(cm.shape[0])

    # Label the axes
    ax.set(title="Confusion Matrix",
           xlabel="Predicted label",
           ylabel="True label",
           xticks=np.arange(n_classes), # create enough axis slots for each class
           yticks=np.arange(n_classes),
           xticklabels=labels, # axes will labeled with class names (if they exist) or ints
           yticklabels=labels)

    # Make x-axis labels appear on bottom
    ax.xaxis.set_label_position("bottom")
    ax.xaxis.tick_bottom()

    ### Changed (plot x-labels vertically) ###
    plt.xticks(rotation = 70, fontsize = text_size)
    plt.yticks(fontsize = text_size)

```

```

### Changed (plot x-labels vertically) ###
plt.xticks(rotation = 70, fontsize = text_size)
plt.yticks(fontsize = text_size)

# Set the threshold for different colors
threshold = (cm.max() + cm.min()) / 2.

# Plot the text on each cell
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    if norm:
        plt.text(j, i, f"{cm[i, j]} ({cm_norm[i, j]*100:.1f}%)",
                  horizontalalignment="center",
                  color="white" if cm[i, j] > threshold else "black",
                  size=text_size)
    else:
        plt.text(j, i, f"{cm[i, j]}",
                  horizontalalignment="center",
                  color="white" if cm[i, j] > threshold else "black",
                  size=text_size)

# Save the figure to the current working directory
if savefig:
    fig.savefig("confusion_matrix.png")

```

Bu kod, bir karışıklık matrisi (confusion matrix) oluşturmayı sağlayan bir fonksiyonu tanımlar. Karışıklık matrisi, modelin tahminlerinin gerçek etiketlerle karşılaştırılmasını görselleştirir ve modelin sınıflandırma performansını değerlendirmek için kullanılır

```
# Plot the confusion matrix using our function
make_confusion_matrix(y_true = yLabels,
                      y_pred = predClasses,
                      classes = class_names,
                      figsize = (10, 10),
                      text_size = 10)
```

Bu kod, önceden tanımladığınız `make_confusion_matrix` fonksiyonunu kullanarak karışıklık matrisini görselleştirmektedir. Fonksiyonu, gerçek etiketlerin `yLabels` ve modelin tahminlerinin `predClasses` olduğu bir karışıklık matrisi oluşturmak için çağırıyoruz. Ayrıca sınıf adlarını (`class_names`), grafiğin boyutunu (`figsize`) ve metin boyutunu (`text_size`) belirtiyoruz.

```
# Get all of the image file paths in the test dataset
filepaths = []
for filepath in test_data.list_files('/content/drive/MyDrive/TUBITAK_EYUP/NEU/test/*/*.bmp',
                                   shuffle = False): # the first * means every subdirectory in test and the second *.jpg means every jpg image in the subdirectory.
    filepaths.append(filepath.numpy())

# Create a DataFrame of different parameters for each of our test images
import pandas as pd
pred_df = pd.DataFrame({'img_path': filepaths,
                       'y_true': yLabels,
                       'y_pred': predClasses,
                       'pred_conf': predProbs.max(axis = 1), # get the maximum prediction probability
                       'y_true_classname': [class_names[i] for i in yLabels],
                       'y_pred_classname': [class_names[i] for i in predClasses]})

# Find out in our DataFrame which predictions are wrong
pred_df['pred_correct'] = pred_df['y_true'] == pred_df['y_pred']

# Sort our DataFrame to have most wrong predictions at the top
top_100_wrong = pred_df[pred_df['pred_correct'] == False].sort_values('pred_conf', ascending = False)[:100]

# Visualize the test data samples which have the wrong prediction but highest prediction probability
images_to_view = 9
start_index = 0
plt.figure(figsize = (15, 10))
for i, row in enumerate(top_100_wrong[start_index: start_index + images_to_view].itertuples()):
    plt.subplot(3, 3, i + 1)
    img = load_and_prep_image(row[1], scale = False)
    _, pred_prob, y_true_classname, y_pred_classname, _ = row # only interested in a few parameters of each row
    plt.imshow(img/255.)
    plt.title(f'Actual: {y_true_classname}, Pred: {y_pred_classname} \n Prob: {pred_prob}')
    plt.axis(False)
```

Bu kod, test veri setinde yanlış tahmin yapılan ve en yüksek tahmin olasılığına sahip olan görüntüleri görselleştirmektedir. İlk olarak, test veri setindeki tüm görüntü dosya yollarını `filepaths` listesine ekliyoruz. Ardından, `pred_df` adında bir DataFrame oluşturuyoruz ve her bir test görüntüsü için çeşitli parametreleri içeriyor. Bu parametreler arasında görüntü dosya yolu, gerçek etiket (`y_true`), tahmin edilen etiket (`y_pred`), tahmin edilen sınıfın olasılığı (`pred_conf`), gerçek etiketin sınıf adı (`y_true_classname`) ve tahmin edilen etiketin sınıf adı (`y_pred_classname`) yer almaktadır. Sonra, DataFrame üzerinde yanlış tahmin yapılan örnekleri belirlemek için `pred_correct` sütunu oluşturuyoruz. DataFrame'i `pred_conf` sütununa göre en yüksek olasılığa sahip olan en yüksek 100 yanlış tahmini seçiyoruz. En yüksek 100 yanlış tahminli örneği görselleştirmek için `images_to_view` değişkeninde belirlediğimiz sayıda örneği gösteriyoruz. Başlangıç indeksini `start_index` ile belirleyebiliriz. Her bir örneği `plt.subplot` fonksiyonuyla ayrı ayrı çizdiriyoruz ve gerçek etiket, tahmin edilen etiket ve tahmin olasılığını başlık olarak ekliyoruz.

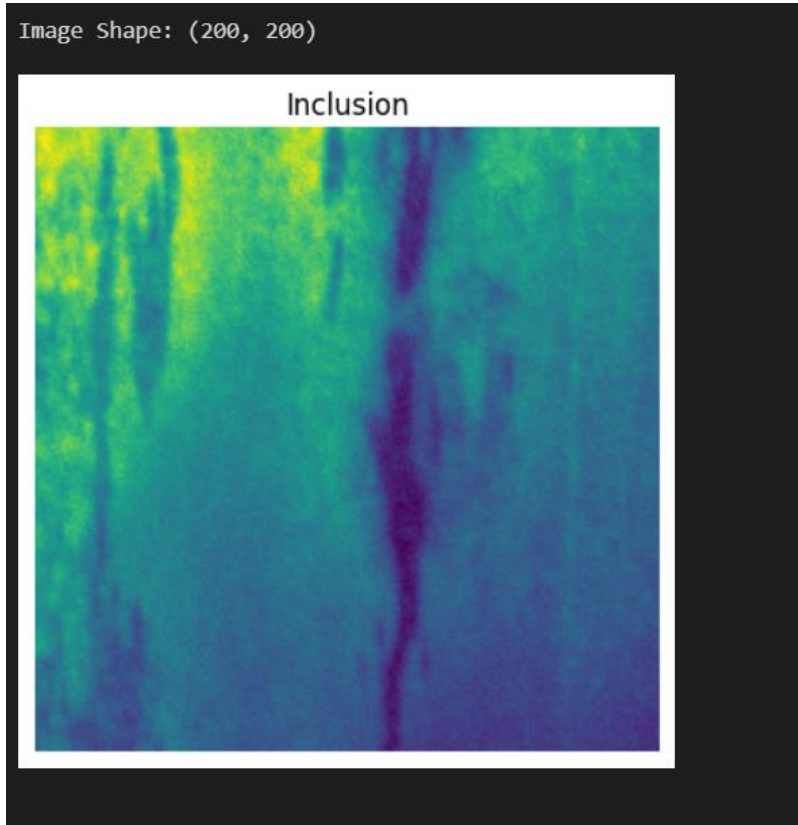


### 3.1.Çıktılar

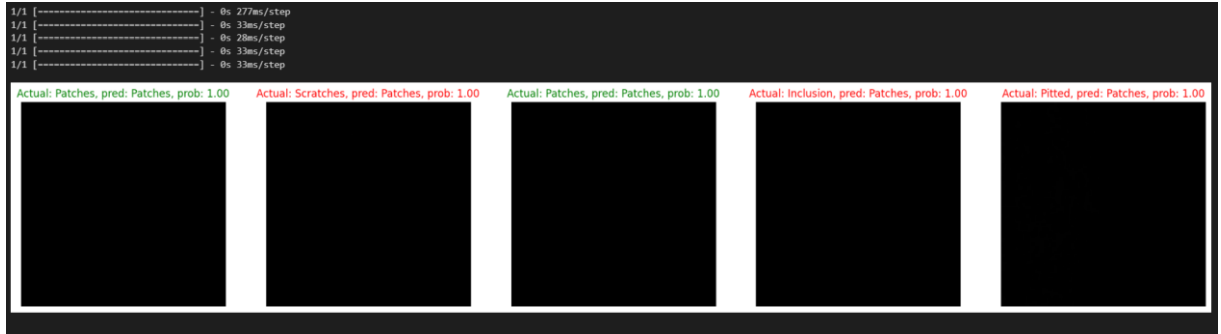


Yukarıdaki Grafiklerde Eğitim kayıpları, Validasyon kayıpları, Training Doğruluğu ve Validasyon Doğruluğu görselleştirilmiştir. Grafiklere göre:

Traning loss (Eğitim Kayıp) değeri 68.9583 den başlayarak 52 epoch sonunda 0.1896 ya yaklaşmıştır. Traning Accuracy(Eğitim doğrulama) değeri, 0.2156 dan başlayarak 0.9088 e yükselmiştir. Validation loss(Validasyon kayıp) değeri 2.0085 dan 0.7189' a yaklaşmıştır. Validation Accuracy(Validasyon Doğrulama) değeri 0.5417'den 0.8194 'e yükselmiştir.

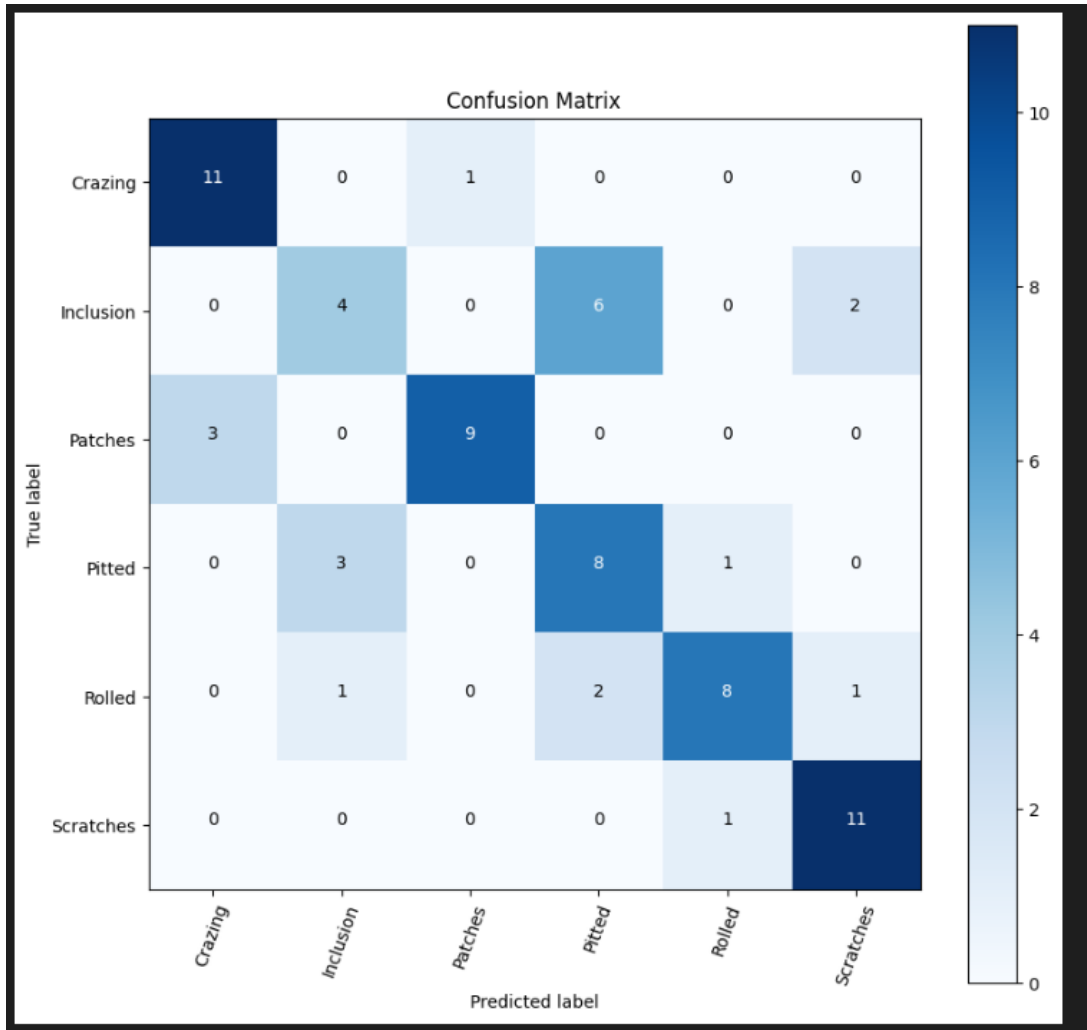


Projede oluşturulmuş olan viewRandomImage() fonksiyonunu ile Projede kullanılmakta olan veri setinden rastgele bir resim görselleştirilmiştir.



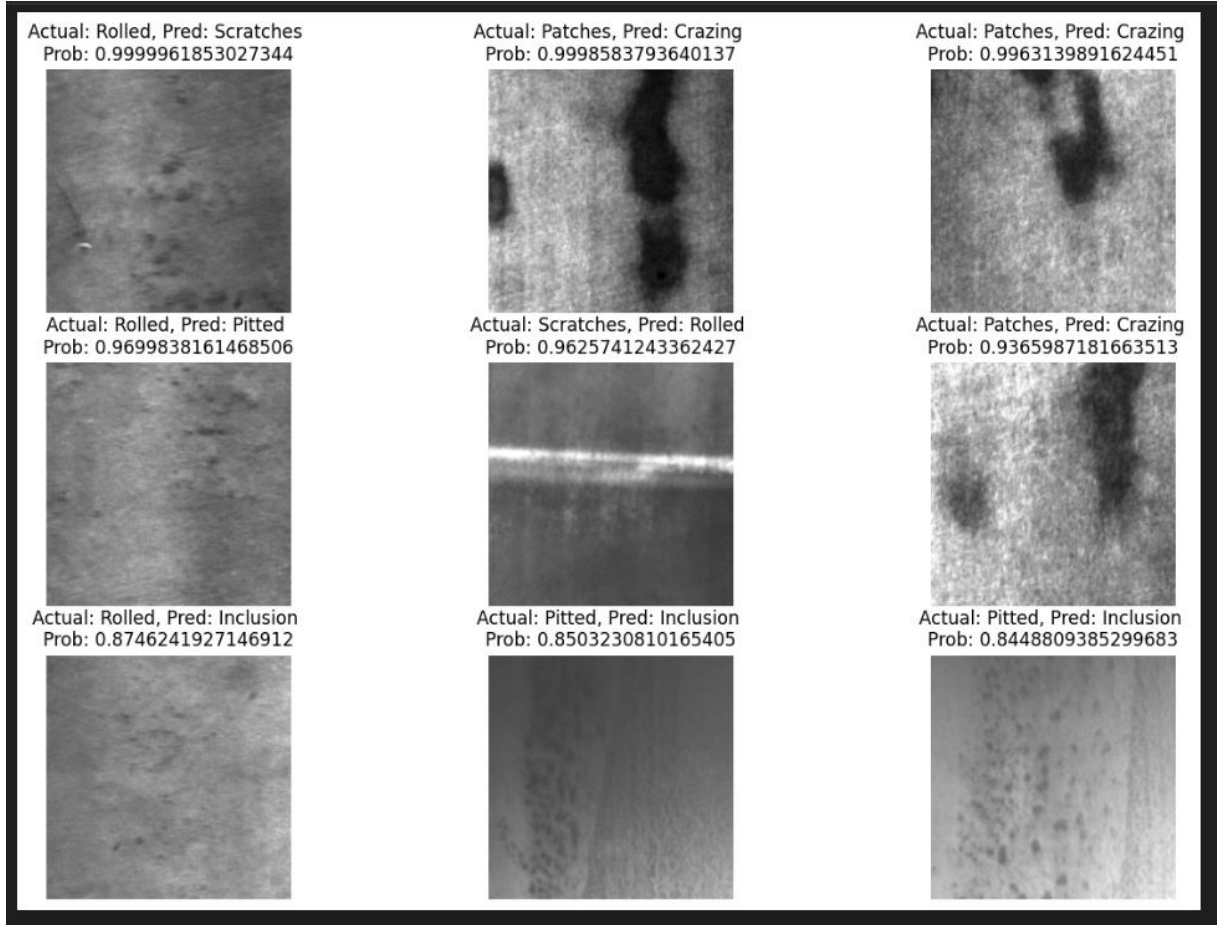
Veri setinde bulunan Test verilerine göre yapılan tahmin edilen sınıflar ve gerçek sınıflar karşılaştırıp görselleştirilmiştir.

Veri setinde bulunan görseller. bmp uzantılı olduğu için görseller matplotlib tarafından görselleştirilememektedir



Yukarıdaki görselde projemizde oluşturduğumuz model'in çıktıları baz alınarak confusion matrix çizdirilmiştir.

Buradan modelin true positive true negative false possitife ve false negative değerleri çıktı olarak alınmıştır.



Kullanılan veri setinde modelin öğrenmesinde en çok karışıklığa sebep olan sınıflar ve karıştırıldığı sınıflar görselleştirilmiştir.

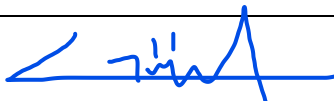
### 3.2.Proje ile ilgili harcama kalemleri hakkında ayrıntılı bilgi

Sarf malzemesi	Fiyat
JetsonNano Ekran Kartı	4000 TL
<b>Toplam Fiyat</b>	<b>4000TL</b>

### KAYNAKLAR

- [1] Q. Luo, X. Fang, L. Liu, C. Yang and Y. Sun, "Automated Visual Defect Detection for Flat Steel Surface: A Survey," in IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 3, pp. 626-644, March 2020, doi: 10.1109/TIM.2019.2963555.
- [2] Di, H., Ke, X., Peng, Z., & Dongdong, Z. (2019). Surface defect classification of steels with a new semisupervised learning method. Optics and Lasers in Engineering, 117, 40-48.
- [3] Liu, K., Wang, H., Chen, H., Qu, E., Tian, Y., & Sun, H. (2017). Steel surface defect detection using a new Haar–Weibull-variance model in unsupervised manner. IEEE transactions on instrumentation and measurement, 66(10), 2585-2596.

- [4] D. He, K. Xu and P. Zhou, "Defect detection of hot rolled steels with a new object detection framework called classification priority network", Comput. Ind. Eng., vol. 128, pp. 290-297, Feb. 2019.
- [5] Dong, H., Song, K., He, Y., Xu, J., Yan, Y., & Meng, Q. (2019). PGA-Net: Pyramid feature fusion and global context attention network for automated surface defect detection. IEEE Transactions on Industrial Informatics, 16(12), 7448-7458.
- [6] H. Wang, J. Zhang, Y. Tian, H. Chen, H. Sun and K. Liu, "A simple guidance template-based defect detection method for strip steel surfaces", IEEE Trans. Ind. Informat., vol. 15, no. 5, pp. 2798-2809, May 2019ç
- [7] Luo, Q., Sun, Y., Li, P., Simpson, O., Tian, L., & He, Y. (2018). Generalized completed local binary patterns for time-efficient steel surface defect classification. IEEE Transactions on Instrumentation and Measurement, 68(3), 667-679.
- [8] He, Y., Song, K., Meng, Q., & Yan, Y. (2019). An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. IEEE Transactions on Instrumentation and Measurement, 69(4), 1493- 1504.
- [9] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation", IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481-2495, Dec. 2017.
- [10] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid scene parsing network", Proc. IEEE Conf. Comput. Vision Pattern Recognit., pp. 6230-6239, 2017.
- [11] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets atrous convolution and fully connected CRFs", IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 834-848, Apr. 2018.
- [12] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation", Proc. IEEE Conf. Comput. Vision Pattern Recognit., pp. 3431-3440, 2015.
- [13] G. Lin, A. Milan, C. Shen and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation", Proc. IEEE Conf. Comput. Vision Pattern Recognit., pp. 5168-5177, 2017.
- [14] Fu, G., Sun, P., Zhu, W., Yang, J., Cao, Y., Yang, M. Y., & Cao, Y. (2019). A deep-learning-based approach for fast and robust steel surface defects classification. Optics and Lasers in Engineering, 121, 397- 405.
- [15] L. Song, W. Lin, Y.-G. Yang, X. Zhu, Q. Guo and J. Xi, "Weak microscratch detection based on deep convolutional neural network", IEEE Access, vol. 7, pp. 27547-27554, 2019. [16] F. Zhou, G. Liu, F. Xu and H. Deng, "A generic automated surface defect detection based on a bilinear model", Appl. Sci., vol. 9, no. 15, pp. 3159, Aug. 2019.
- [17] [http://faculty.neu.edu.cn/yunhyan/NEU\\_surface\\_defect\\_database.htm](http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.htm)

PROJE YÜRÜTÜCÜSÜNÜN ADI – SOYADI - İMZA	DANIŞMANIN ADI – SOYADI - İMZA
Eyüphan Şahin	Dr. Öğrt. Üyesi Gülşah Karaduman
	

Tarih : 05.06.2023