**MRM3048.1 –****Mekatronik Sistem Tasarımı**

***Ödev Raporu***

**Omar Tarabin 170221992**

**Görüntü İşleme ile Saç Levha Üzerindeki Yüzey Hataları Tespiti**

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

1. **Sac metalde yüzey kusurlarinin tespi̇ti̇ni̇n önemi̇**

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 hurda 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.

1. **Bazı yüzey kusurları**

Bu çalışmada NEU Metal Yüzey Kusurları veri seti kullanılmıştır. Bu veri seti, sıcak haddelenmiş çelik şeritlerin altı tipik yüzey kusur fotoğraf örneklerini içermektedir: haddelenmiş ölçek, yamalar, çentiklenme, çukurlu yüzey, inklüzyon ve çizikler. Veritabanı, her altı tipik yüzey kusurundan 300 fotoğraf olmak üzere toplamda 1800 fotoğraf içerir. Bu veri kümesi altı farklı kusur türünün görüntülerini içermektedir:

* 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:

A collage of different images of different shapes

Description automatically generated with medium confidence

1. **Yapay Sinir Ağları ve CNN**

Yapay sinir ağları birbirine bağlı yapay nöron denen zımbırtıların 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.

A diagram of a diagram

Description automatically generated

CNN (Evrişimli Sinir Ağı), yaygın olarak görsel görüntüleri analiz etmek için uygulanan bir yapay sinir ağı sınıfıdır. Bir girdi olarak bir görüntü aldıktan sonra onu çeşitli katmanlardan geçirerek görüntüdeki farklı özelliklere değer verir (ağırlık verir) ve sınıflandırma yapar.

![A diagram of a network

Description automatically generated]()

1. **Metot ve Uygulama**

Bu kısımda sınıflandırma için yapılan uygulamanın metodu detaylı olarak adım adım, bazen de kodlama örnekleri ile birlikte anlatılacaktır.

* Çalışma Ortamın Ayarlanması: Bu adım, çalışma ortamı için gerekli kütüphanelerin ve bağımlılıkların yüklenmesini ve varsa donanımın (örn. GPU, CPU) 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 ayrılır. 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), eğitim sırasında modeli çeşitli veri varyasyonlarına maruz bırakarak modelin genelleme yeteneğini geliştirmeye yardımcı olur. Bu, modelin bu dönüşümlere karşı değişmez olan özellikleri öğrenmesine ve girdi verilerindeki değişikliklere karşı daha dayanıklı olmasına yardımcı olabilir.
* Model Oluşturma: Bu raporda, 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.
* Model Eğitme: Model, eğitim için uygun sparse categorical crossentropy kayıp fonksiyonu ve optimize edici Adam ile derlenmiştir. Eğitim süreci, mini gruplar halinde eğitim seti üzerinde yineleme, kaybı hesaplama ve geriye yayılım kullanarak modelin ağırlıklarını güncellemeyi içeriyordu. Model, aşırı uyumu önlemek için erken durdurma ile belirli sayıda epok için eğitilmiştir.
* Değerlendirme: Eğitimden sonra model, doğruluk, kesinlik, geri çağırma ve F1-skoru gibi ölçütler kullanılarak performansını değerlendirmek için doğrulama kümesi üzerinde değerlendirilmiştir. Son olarak model, görülmeyen veriler üzerindeki performansını değerlendirmek ve genelleme yeteneğini ölçmek için ayrı bir test kümesi üzerinde test edilmiştir.

Raporun bu kısmında Python ve TensorFlow kullanarak modelin oluşturulmasını, eğitilmesini ve sonuçlarını analizini içeren kodun her parçası detaylı olarak anlatılacaktır.

And this does this and that. The following snippet will loop through and show some of the images in the dataset along with their labels. Veri içerisinde bulunan bazı görseller aşağıdaki gibidir:

|  |
| --- |
| input\_shape = (BATCH\_SIZE, IMAGE\_SIZE, IMAGE\_SIZE, CHANNELS)  n\_classes = len(class\_names)  model = models.Sequential([  resize\_and\_rescale,  layers.Conv2D(32, kernel\_size = (3,3), activation='relu', input\_shape=input\_shape),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, kernel\_size = (3,3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, kernel\_size = (3,3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D((2, 2)),  layers.Flatten(),  layers.Dense(64, activation='relu'),  layers.Dense(n\_classes, activation='softmax'),  ])  model.build(input\_shape=input\_shape)  model.compile(  optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),  loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=False),  metrics=['accuracy']  ) |

This is the fun bit. Include a picture that shows the structure of the neural network so it looks cool.

|  |
| --- |
| 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')]  ) |

'model.fit()' fonkisyonu modeli eğitme sürecini başlatır. Burada 15 döngü boyunda, 45 adet batch’te bulunan tüm verileri işleyerek model kendi parametrelerini loss’u minimum tutacak şekilde günceller ve validation veri grubu ile doğrulama yapar. Model eğitilirken aşağıdaki çıktıyı sürekli verir ve görüldüğü her epoch’ta modelin kaybı yavaşça 0’ya yaklaşmakta, aynı zamanda doğruluk oranı da 1’e yaklaşmaktadır.

1. **Sonuçlar ve Doğrulama**

A graph of loss and loss

Description automatically generated

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. Bu iki grafik şu şu şu yüzden önemli ve performans metrik olarak kullanılır.

Doğrulama için basit bir bilmem ne yapılmış sonra daha detaylı bir inceleme için bilmem ne fonkisyonu yazılmış.

|  |
| --- |
| scores = model.evaluate(test\_ds) |

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

Görüldüğü gibi modelimiz test veri grubu üzerinde 0,9312 doğrulama oranı ile tahmin yapabilmektedir. Bu tahmin doğruluğunu arttırmak icin farkli bir model mimarisi denenebilir veya daha fazla veri daha uzun döngüler boyunce eğitilebilir. Son olarak aşağıdaki kısım test verisi üzerinde tek tek fotoğraflar üzerinde model tahmini yapılarak fotoğrafın kendisi gerçek sonuç ve tahmin ile birlikte gösterilmiştir.

A collage of different images

Description automatically generated