

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ



MAKİNE ÖĞRENMESİ İLE KUMAŞ HATASI TESPİTİ

LİSANS BİTİRME ÇALIŞMASI

Birce Ceren PINAR
Turhan GEZER

Bilgisayar Mühendisliği Bölümü

TEZİN SAVUNULDUĞU AY YIL

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ



MAKİNE ÖĞRENMESİ İLE KUMAŞ HATASI TESPİTİ

LİSANS BİTİRME ÇALIŞMASI

Turhan GEZER
(18360859061)

Birce Ceren PINAR
(18360859015)

Bilgisayar Mühendisliği Bölümü

Danışman: Doç. Dr. Haydar ÖZKAN

HAZİRAN, 2023

BTÜ, Mühendislik ve Doğa Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü'nün 18360859015 - 18360859061 numaralı öğrencileri Birce Ceren PINAR ve Turhan GEZER, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "MAKİNE ÖĞRENMESİ İLE KUMAŞ HATASI TESPİTİ" başlıklı bitirme çalışmasını aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Danışmanı : **Doç. Dr. Haydar ÖZKAN**
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Dr. Öğr. Üyesi Adı SOYADI**
Bursa Teknik Üniversitesi

Öğr. Gör. Dr. Adı SOYADI
Bursa Teknik Üniversitesi

Savunma Tarihi : 15 HAZİRAN 2023



BM Bölüm Başkanı : Prof. Dr. Turgay Tugay Bilgin
Bursa Teknik Üniversitesi/...../.....

İNTİHAL BEYANI

Bu bitirme alışmasında grsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, bitirme alışması içinde yer alan ancak bu alışmaya özgü olmayan tüm sonuç ve bilgileri bitirme alışmasında kaynak göstererek belgelediğimi, aksinin ortaya ıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Birce Ceren PINAR
Turhan GEZER

İmzası :

ÖNSÖZ

Tekstil endüstrisi, günümüzde büyük bir öneme sahip olan ve pek çok alanda kullanılan kumaşların üretiminde önemli bir rol oynamaktadır. Ancak, kumaş üretimi sürecindeki hatalar, hem üreticiye hem de son kullanıcıya olumsuz etkileri olan bir sorun haline gelebilir. Bu hatalar, ürün kalitesinin düşmesine, geri çağırma maliyetlerinin artmasına ve müşteri memnuniyetinin azalmasına neden olabilir. Dolayısıyla, kumaş üretiminde kalite kontrolünün etkin bir şekilde yapılması büyük bir önem taşımaktadır.

Bu proje, bilgisayar görüşü ve derin öğrenme tekniklerinin kullanımını içeren bir araştırma çalışmasını sunmaktadır. Amacımız, kumaş üretimi sürecindeki kalite kontrolünü geliştirmek ve kumaş hatalarının otomatik olarak tespitini sağlamaktır. Geleneksel yöntemlerle hata tespiti genellikle zaman alıcı, maliyetli ve doğruluk düzeyi düşük olabilmektedir. Bu nedenle, bilgisayar görüşü ve derin öğrenme tekniklerinin entegrasyonu ile daha doğru, hızlı ve etkili bir hata tespit sistemi oluşturmayı hedefliyoruz.

Bu proje kapsamında, bir bilgisayar kamerası kullanarak kumaşların görüntülerini elde edeceğiz. Veri toplama aşamasında, hatalı ve hatasız kumaş örneklerini içeren bir veri seti oluşturacağız. Bu veri seti, derin öğrenme modelimizin eğitimi ve testi için kullanılacaktır. Veri ön işleme adımlarıyla görüntüleri düzenleyecek, özellik çıkarma yöntemleriyle anlamlı özelliklerin çıkarılmasını sağlayacak ve derin öğrenme modelini tasarlayıp eğiteceğiz. Sonuç olarak, gerçek zamanlı bir kumaş kontrol sistemi geliştirerek, kumaş hatalarının otomatik olarak tespit edilmesini ve üretim sürecinin iyileştirilmesini hedefliyoruz.

Bu projeyi tamamlamak için yoğun bir çalışma ve işbirliği gerektiğinin bilincindeyiz. Ancak, elde edeceğimiz sonuçların kumaş endüstrisi için önemli bir katkı sağlayacağına inanıyoruz. Bu çalışma, üretkenliği artırarak maliyetleri azaltmak, ürün kalitesini yükseltmek ve müşteri memnuniyetini artırmak için bir adım olacaktır.

Bu projeyi tamamlamamızda bize destek olan herkese teşekkür ederiz. Özellikle danışmanımızın yönlendirmeleri ve teşvikleri için ayrıca minnettarız. Umarız bu çalışma, kumaş üretimi ve kalite kontrolü konusunda ilgili araştırmacılar ve endüstri profesyonelleri için faydalı bir kaynak olacaktır.

Haziran 2023

Birce Ceren PINAR – Turhan GEZER

İÇİNDEKİLER

Sayfa

ÖNSÖZ	v
İÇİNDEKİLER	vi
KISALTMALAR	vii
ŞEKİL LİSTESİ	viii
ÖZET	ix
SUMMARY	x
1. GİRİŞ	11
1.1 Tezin Amacı	11
1.2 Literatür Araştırması	12
1.3 Hipotez	14
2. KULLANILAN TEKNOLOJİLER	15
2.1 Python	15
2.1.1 PyTorch	16
2.1.2 OpenCV	16
2.1.3 Flask	17
2.1.4 Matplotlib	17
2.2 .NET Framework	17
2.3 C#	18
2.4 Google Colab	19
3. METODOLOJİ	20
3.1 Proje Geliştirme Süreci	20
3.2 Proje Taslağının Oluşturulması	21
3.3 Araçların Kurulumu ve Yönetimi	21
3.4 Kamera Arayüzü	22
3.5 Verilerin Toplanması	28
3.6 Verilerin Sınıflandırılması	29
3.7 Modelin Eğitimi	30
3.8 Modelin performansı	36
3.9 API Oluşturulması	41
4. UYGULAMA ÇIKTILARI	42
4.1 Uygulamanın Arayüzü	42
4.2 Kamera	42
4.3 Fotoğraf Yükleme	43
4.4 Çalıştır	44
5. SONUÇ	46
5.1 Çalışmanın Uygulama Alanı	47
6. KAYNAKLAR	48

KISALTMALAR

API	: Application Programming Interface
CNN	: Convolutional Neural Network
ML	: Machine Learning
OOP	: Object-Oriented Programming
PY	: Python
GPU	: Graphics Processing Unit
HTTP	: Hypertext Transfer Protocol
CUDA	: Compute Unified Device Architecture
IDE	: Integrated Development Environment

ŞEKİL LİSTESİ

	Sayfa
Şekil tablosu ögesi bulunamadı.	
Şekil 3.1 : Kamera Arayüz Tasarımı	23
Şekil 3.2 : Kamera Başlangıç Butonu Fonksiyonu	23
Şekil 3.3 : Fotoğraf Yakalama Butonu Fonksiyonu	25
Şekil 3.4 : Fotoğraf Yükleme Butonu Fonksiyonu	26
Şekil 3.5 : 'Çalıştır' Butonu fonksiyonu	27
Şekil 3.6 : Makesense.ai üzerinde veri etiketleme	29
Şekil 3.7 : Etiketlenmiş bir görselin sınıf ve koordinatlarını içeren .txt dosyası	29
Şekil 3.8 : Modeller ver Performans Metrikleri	30
Şekil 3.9 : Sınıfların belirtildiği Custom.yaml dosyası	31
Şekil 3.10 : 'Hole/Cut' sınıfı örnek görsel	31
Şekil 3.11 : 'Stain' sınıfı örnek görsel	32
Şekil 3.12 : Yolov7'yi GitHub'tan klonlamak ve Model eğitimi	32
Şekil 3.13 : Detect.py dosyasına gerekli parametreleri göndermek ve çalıştırmak ...	34
Şekil 3.14 : Test Görseli, model kumaştaki deliği tespit etti	35
Şekil 3.15 : Test görseli, Model Kumaş üzerindeki lekeyi tespit etti	35
Şekil 3.16 : Precision ve Recall Metrikleri. X eksenİ İterasyon sayısıdır.....	36
Şekil 3.17 : Recall - Confidence Grafiği	37
Şekil 3.18 : Precision - Recall Grafiği	37
Şekil 3.19 : Precision - Confidence Grafiği	38
Şekil 3.20 : F1 - Confidence Grafiği	39
Şekil 3.21 : Karmaşıklık Matrisi	40
Şekil 3.22 : Python'ı localhost'ta çalıştıran flask fonksiyonu	41
Şekil 4.1 : Uygulama Arayüzü	42
Şekil 4.2 : Kameraya gösterilen bir kumaş görüntüsü	42
Şekil 4.3 : Kamera seçme menüsü	43
Şekil 4.4 : Soldaki görsel Fotoğraf çek butonu ile kaydedilip sağdaki kutuda görüntüleniyor	43
Şekil 4.5 : Gözet penceresi	44
Şekil 4.6 : Seçilen Görsel	44
Şekil 4.7 : İşlem Başarılı ise Gelen Mesaj Kutusu	45
Şekil 4.8 : İşlem Başarısız ise gelen Mesaj Kutusu	45
Şekil 4.9 : API'den başarı ile dönen görsel	45

MAKİNE ÖĞRENMESİ İLE KUMAŞ HATASI TESPİTİ

ÖZET

Bu projenin amacı, tekstil endüstrisinde kumaş hatalarının otomatik olarak tespit edilmesini sağlayan bir sistem geliştirmektir. Geleneksel yöntemlerle yapılan kalite kontrol süreçleri zaman alıcı, maliyetli ve hata eğilimli olabilir. Bu nedenle, bilgisayar görüşü ve derin öğrenme tekniklerini kullanarak daha doğru, hızlı ve etkili bir hata tespit sistemi oluşturmayı hedefliyoruz.

Projede, bir bilgisayar kamerası kullanarak kumaşlardan yüksek çözünürlüklü görüntüler elde edilecek. Bu görüntüler üzerinde veri toplama ve ön işleme adımları gerçekleştirilecek. Hatalı ve hatasız kumaş örneklerini içeren bir veri seti oluşturulacak ve derin öğrenme modeli bu veri setiyle eğitilecektir. Model, kumaş örneklerinden özellikler çıkaracak ve hatalı bölgeleri tespit etmek için kullanılacaktır.

Bu proje başarıyla tamamlandığında, kumaş üretimindeki kalite kontrol süreçleri iyileştirilecek ve hatalar otomatik olarak tespit edilecektir. Gerçek zamanlı olarak kullanılabilen bu otomatik hata tespit sistemi, üretim süreçlerinde hataların erken tespit edilmesini sağlayacak ve üretkenliği artıracaktır. Ayrıca, marka itibarının korunması ve müşteri memnuniyetinin artırılması için kalite kontrol standartlarının yükseltilmesine katkı sağlayacaktır. Bu proje, kumaş endüstrisindeki kalite kontrol süreçlerinin geliştirilmesi ve bilgisayar görüşü ile derin öğrenme tekniklerinin diğer alanlarda da kullanılması konusunda önemli bir adım olarak değerlendirilmektedir.

Anahtar kelimeler: kumaş, kalite kontrol, hata tespiti, otomatik, bilgisayar görüşü, derin öğrenme.

FABRIC DEFECT DETECTION USING MACHINE LEARNING

SUMMARY

The aim of this project is to develop a system that automatically detects fabric defects in the textile industry. Traditional methods of quality control in fabric production can be time-consuming, costly, and prone to human error. Therefore, our objective is to create a more accurate, fast, and effective defect detection system by integrating computer vision and deep learning techniques.

In this project, we will use a computer camera to capture high-resolution images of fabrics. Data collection and preprocessing steps will be performed on these images. A dataset containing both defective and defect-free fabric samples will be created, and a deep learning model will be trained using this dataset. The model will extract features from fabric samples and be used to detect faulty regions.

Upon successful completion of this project, the quality control processes in fabric production will be improved, and defects will be automatically detected. The real-time automated defect detection system can facilitate early detection of errors in production processes, thereby enhancing productivity and reducing recall costs. Additionally, it will contribute to elevating quality control standards, ensuring customer satisfaction, and maintaining brand reputation. This project is considered a significant step in enhancing quality control processes in the textile industry and inspiring the application of computer vision and deep learning techniques in other domains.

Keywords: fabric, quality control, defect detection, automatic, computer vision, deep learning.

1. GİRİŞ

Tekstil endüstrisi, günlük yaşantımızda önemli bir yer tutan giyimden ev tekstiline kadar birçok alanda kullanılan kumaşların üretiminde büyük bir rol oynamaktadır. Kumaşların üretim sürecindeki kalite kontrolü, hem ürün kalitesinin belirlenmesinde hem de müşteri memnuniyetinin sağlanmasında kritik bir faktördür. Ancak, geleneksel olarak manuel olarak yapılan kalite kontrol süreçleri zaman alıcı, maliyetli ve insan hatasına açık olabilir. Bu nedenle, otomatik kumaş hata tespiti sistemlerinin geliştirilmesi, kumaş endüstrisinde verimliliği artırmak ve hatalı ürünlerin pazara sürülmesini engellemek için son derece önemli bir adımdır.

1.1 Tezin Amacı

Bu projenin amacı, gelişmiş bilgisayar görüşü ve derin öğrenme tekniklerini kullanarak kumaş hatalarını otomatik olarak tespit eden bir sistem geliştirmektir. Geleneksel yöntemlerle yapılan kumaş hata tespiti süreçleri, insan faktörüne bağlı hatalar, düşük doğruluk ve yüksek maliyet gibi zorluklarla karşılaşabilir. Bu nedenle, bilgisayar görüşü ve derin öğrenme tekniklerini entegre ederek daha doğru, hızlı ve etkili bir hata tespit sistemi oluşturmayı hedeflemektedir.

Bu projede, bilgisayar kamerası kullanarak yüksek çözünürlüklü görüntüler elde edeceğiz. Farklı kumaş türlerinden alınan örnekler üzerinde veri toplama ve ön işleme adımları gerçekleştireceğiz. Hatalı ve hatasız kumaş örneklerini içeren bir veri seti oluşturarak, derin öğrenme modelini eğiteceğiz. Modelimiz, kumaş örneklerinden özellikleri çıkaracak ve hatalı bölgeleri tespit etmek için kullanılacaktır.

Bu projenin başarıyla tamamlanmasıyla, kumaş üretimindeki kalite kontrol süreçleri iyileştirilecek ve hataların otomatik olarak tespit edilmesi sağlanacaktır. Gerçek zamanlı kullanılabilen bu otomatik hata tespit sistemi, üretim süreçlerindeki hataların erken tespit edilmesini sağlayarak geri çağırma maliyetlerini azaltacak ve üretkenliği artıracaktır. Aynı zamanda, müşteri memnuniyetini artırarak marka itibarını koruma

ve rekabet avantajı elde etme amacıyla kalite kontrol standartlarının yükseltilmesine de katkı sağlayacaktır.

1.2 Literatür Araştırması

Wu, Y., Wang, X., Li, Y., & Li, X. (2020). "Fabric defect detection using deep convolutional neural network with pyramid pooling and hard sample mining." Optik, 213, 164666. Bu çalışma, derin evrişimli sinir ağı (CNN) kullanarak kumaş hata tespiti için bir yöntem önermektedir. Makalede, piramit havuzlama ve zorlu örnek madenciliği tekniklerini içeren bir CNN modeli tanıtılmaktadır. Piramit havuzlama, farklı ölçeklerdeki özniteliklerin dikkate alınmasını sağlar, böylece farklı boyutlardaki hataların etkin bir şekilde tanınmasına olanak tanır. Zorlu örnek madenciliği, modelin hatalı bölgeleri daha iyi öğrenmesine yardımcı olmak için eğitim sürecindeki zorlu örnekleri vurgular.

Makalede, kumaş hata tespiti için kullanılan bir veri seti üzerinde deneyler gerçekleştirilmiştir. Önerilen yöntem, diğer yaygın kullanılan yöntemlerle karşılaştırıldığında daha yüksek bir hata tespit başarısı göstermiştir. Ayrıca, modelin zorlu örnek madenciliği teknikleriyle eğitilmesi, hata tespiti performansını daha da artırmıştır.

Sonuç olarak, bu çalışma, piramit havuzlama ve zorlu örnek madenciliği gibi tekniklerin kullanıldığı derin CNN modellerinin, kumaş hata tespiti alanında etkili bir yaklaşım olduğunu göstermektedir. Bu tür yöntemler, kumaş endüstrisindeki kalite kontrol süreçlerini geliştirmek ve hatalı ürünlerin tespitini daha hassas hale getirmek için potansiyel sağlamaktadır.

Zhang, Y., Zhang, J., & Wu, D. (2020). "Fabric defect detection based on improved faster R-CNN." Journal of Textile Research, 41(02), 35-41. Bu makale, geliştirilmiş bir Faster R-CNN (Region-Based Convolutional Neural Network) kullanarak kumaş hata tespiti için bir yöntem sunmaktadır. Faster R-CNN, nesne tespiti ve sınıflandırma için popüler bir derin öğrenme modelidir. Makalede, Faster R-CNN modeline bazı iyileştirmeler yapılmış ve kumaş hata tespiti için uygun hale getirilmiştir.

Yöntem, önceden eğitilmiş bir derin evrişimli sinir ağı (CNN) modeli kullanarak öznitelik çıkarımını gerçekleştirir ve ardından tespit aşamasında bölgeleri sınıflandırır

ve hatalı bölgeleri belirler. Makalede, önerilen yöntemin kumaş hata tespiti üzerindeki performansı deneysel olarak değerlendirilmiştir.

Sonuçlar, geliştirilmiş Faster R-CNN yönteminin kumaş hata tespiti konusunda diğer yöntemlere kıyasla daha yüksek bir doğruluk sağladığını göstermektedir. Yöntem, farklı hata tiplerini başarıyla tespit edebilmekte ve yanlış pozitiflerin sayısını azaltmaktadır. Bu, kumaş endüstrisindeki kalite kontrol süreçlerinin iyileştirilmesi ve hatalı ürünlerin erken tespiti için önemli bir adım olabilir.

Makalede ayrıca, geliştirilen yöntemin bilgisayar kaynakları açısından verimli olduğu ve gerçek zamanlı uygulamalar için uygulanabilir olduğu vurgulanmaktadır. Bu, kumaş üretimindeki kalite kontrol süreçlerinin otomatikleştirilmesi ve verimliliğin artırılması açısından önemli bir avantaj sağlamaktadır.

Güçlüer, Ü., Özyazıcıoğlu, M. N., & Erol, H. (2018). "Real-time fabric defect detection using deep learning approach." IOP Conference Series: Materials Science and Engineering, 433(1), 012047. Bu makale, gerçek zamanlı kumaş hata tespiti için derin öğrenme yaklaşımının kullanıldığı bir yöntem sunmaktadır. Makalede, kumaş endüstrisindeki kalite kontrol süreçlerindeki hataların otomatik olarak tespit edilmesi için bir çözüm önerilmektedir.

Yöntem, derin evrişimli sinir ağları (CNN) kullanılarak gerçekleştirilir. Önceden eğitilmiş bir CNN modeli kullanılarak kumaş örneklerinden özellikler çıkarılır ve ardından bu özellikler kullanılarak hatalı bölgeler tespit edilir. Makalede, yöntemin gerçek zamanlı uygulamalarda başarılı bir şekilde çalıştığı ve hızlı ve doğru sonuçlar sağladığı belirtilmektedir.

Yapılan deneysel çalışmalar, önerilen yöntemin diğer geleneksel yöntemlere kıyasla daha yüksek bir doğruluk oranına sahip olduğunu göstermektedir. Ayrıca, yöntemin hızlı ve verimli bir şekilde çalıştığı ve yüksek kaliteli ürünlerin üretiminde kullanılabilir olduğu vurgulanmaktadır.

Bu makale, kumaş endüstrisindeki kalite kontrol süreçlerinin iyileştirilmesi ve hatalı ürünlerin erken tespiti için derin öğrenme yaklaşımının etkisini göstermektedir. Derin öğrenme teknikleri, kumaş hata tespiti alanında daha doğru ve verimli çözümler sunabilmektedir. Bu, kumaş endüstrisinde verimliliği artırmak, maliyetleri azaltmak ve müşteri memnuniyetini sağlamak için önemli bir adım olabilir.

1.3 Hipotez

Bu proje kapsamında, bilgisayar kamerası kullanılarak kumaş hatalarının otomatik olarak tespit edildiği bir sistem geliştirmeyi amaçlamaktayız. Bu projemiz, derin öğrenme tekniklerinin entegrasyonu ile oluşturulan bu sistem sayesinde, geleneksel yöntemlere kıyasla daha yüksek doğruluk oranları elde edilebileceği ve kumaş hata tespiti süreçlerinin iyileştirilebileceğidir. Ayrıca, bu otomatik hata tespit sisteminin gerçek zamanlı kullanıma uygun olacağı ve üretim süreçlerinde hataların erken tespit edilmesini sağlayarak geri çağırma maliyetlerini azaltabileceği hipotezini kurmaktayız. Bu projenin başarıyla tamamlanmasıyla, kumaş endüstrisindeki kalite kontrol süreçlerinin daha verimli ve etkili hale getirilebileceği ve müşteri memnuniyetinin artırılabilmesi öngörülmektedir.

2. KULLANILAN TEKNOLOJİLER

Günümüzde endüstriyel kameraların çok pahalı olması ve ek olarak bir lense ihtiyaç olması nedeniyle maliyet çok büyümektedir. Bu ihtiyaca yönelik olarak endüstriyel kameralar yerine bilgisayar kameraları kullanılarak, maliyet de bir kazanım elde edilmek istenmektedir. Kameradan elde edilen görüntü, derin öğrenme modeline girdi olarak verilecek ve sonucunda ise sınıflandırma ve hata tespiti yapılması istenecektir. Tüm bu sistem için bir arayüz tasarımı yapılacak ve eğer görüntü üzerinde bir hatalı nokta var ise görüntünün ve hatanın bulunduğu kısım işaretlenerek arayüzde gösterilecektir. Derin öğrenmedeki model eğitim işlemleri Python programlama dilinde yapılacak. Eğitilen modelin ağırlıkları ve modelin kendisi kaydedilerek. Arayüz tasarımının bulunduğu .Net üzerinde kullanılacaktır. Bu verilerin kullanılacağı .Net frameworku ile C# dili kullanılarak bir bilgisayar arayüzü tasarlanılacak. Bu sayede çekilen görüntüler derin öğrenme modeli ile kullanılarak hata tespiti sağlanacaktır.

2.1 Python

Python, yüksek seviyeli, genel amaçlı bir programlama dilidir. Basit ve okunabilir sözdizimiyle öne çıkan Python, öğrenmesi kolay ve hızlı bir şekilde üretkenlik sağlar. İnteraktif kabiliyeti sayesinde kullanıcılarına anında geri bildirim sağlar ve hata ayıklama süreçlerini kolaylaştırır. Python, çeşitli platformlarda (Windows, macOS, Linux) kullanılabilir ve geniş bir kütüphane ekosistemine sahiptir. Bu kütüphaneler, çeşitli görevler için hazır fonksiyonları ve araçları içerir ve Python'u veri analizi, yapay zeka, web geliştirme, bilimsel hesaplama ve daha pek çok alanda kullanılan bir tercih haline getirir. Python, nesne yönelimli programlama (OOP) prensiplerine dayanır ve modüler bir yapıya sahiptir, bu da büyük projelerin yönetimini kolaylaştırır. Ayrıca, açık kaynaklı bir dil olması, Python topluluğunun sürekli büyümesini sağlar ve kullanıcıların birbirleriyle deneyimlerini paylaşmasını kolaylaştırır. Python'ın bu özellikleri, başlangıç seviyesinden uzmanlık seviyesine kadar tüm programcılar için etkili ve verimli bir yazılım geliştirme deneyimi sunar.

2.1.1 PyTorch

PyTorch, derin öğrenme ve makine öğrenme uygulamaları için popüler bir açık kaynaklı Python kütüphanesidir. PyTorch, yüksek seviyede esneklik sunan ve aynı zamanda GPU hızlandırması gibi gelişmiş özellikler sağlayan bir yapay sinir ağı (YSA) kütüphanesidir. Proje kapsamında PyTorch, tekstil ürünlerinin görüntülerini analiz etmek için kullanılan derin öğrenme modelinin geliştirilmesinde temel bir rol oynadı.

PyTorch, grafik işlem birimleri (GPU'lar) üzerinde hızlı hesaplamalar yapabilme yeteneği sayesinde büyük boyutlu görüntü verileri üzerinde etkili bir şekilde çalışmayı sağlar. Bu da tekstil ürünlerinin görüntülerini hızlı bir şekilde işleyerek kusurların tespit edilmesini kolaylaştırır. PyTorch'un esnek ve kullanıcı dostu arayüzü, model eğitimi ve hiperparametre ayarlarını kolaylaştırırken, hızlı prototipleme imkanı sunar.

PyTorch, derin öğrenme için birçok önceden eğitilmiş modelin bulunduğu bir model galerisi sunar. Bu galeriden elde edilen modeller, projenin kusur tespiti aşamasında önemli bir başlangıç noktası olabilir. Aynı zamanda, PyTorch'un grafik işlem birimleri üzerinde paralel hesaplama yeteneklerini kullanarak, bu modellerin tekstil ürünlerinin görüntülerini hızlı bir şekilde değerlendirmesini sağlar.

2.1.2 OpenCV

OpenCV (Open Source Computer Vision Library), açık kaynaklı bir görüntü işleme ve bilgisayar görüşü kütüphanesidir. OpenCV, çeşitli işlevleri ve algoritmaları destekleyerek, görüntü işleme projelerinde geniş bir kullanım alanı bulur. Bu projede, tekstil ürünlerinin görüntülerini analiz etmek için OpenCV'nin bir dizi özelliği ve fonksiyonu kullanıldı.

OpenCV, görüntüleri yükleme, dönüştürme, işleme ve analiz etme gibi temel işlevleri kolaylaştırır. Görüntüler üzerinde yoğun işlemler gerçekleştirmek için optimize edilmiş bir kütüphane olduğundan, büyük boyutlu tekstil görüntülerini verimli bir şekilde işleyebilir. Ayrıca, OpenCV'nin kamera modülü, uygulamanın gerçek zamanlı kamera görüntülerini almasını ve analiz etmesini sağlar.

Proje kapsamında, tekstil ürünlerinin görüntülerinin işlenmesi için OpenCV'nin özellikleri kullanıldı. Örneğin, görüntü düzeltme, segmentasyon, kenar tespiti, kontur analizi ve nesne algılama gibi işlemler OpenCV'nin fonksiyonlarıyla gerçekleştirildi. Bu işlevler, tekstil kusurlarını tespit etmek için önemli bir rol oynar.

2.1.3 Flask

Flask'ı kullanma sebebi C# ile geliştirilmiş arayüzden gelen HTTP isteğini Flask sayesinde almak ve görüntünün dosya konumunu iki program arasında transfer etmektir. Flask, bu senaryoya uygun bir çözüm sağlar. Flask, HTTP isteklerini yönetmek, yönlendirmek ve yanıtlamak için kullanılan birçok özelliği içerir.

Flask, Python ile kolayca entegre olabilen bir çatı olduğundan, C# arayüzünden gelen HTTP isteğini Flask uygulamasına yönlendirmek için kullanılabilir. Flask, C# tarafından yapılan HTTP isteğini alarak içerisindeki dosya konumunu Python kodumuza aktarır. Daha sonrasında Python'ın işlediği görüntüyü C#'a dönüş değeri olarak verir. Bu görüntü C#'ta tasarlanan arayüzde gösterilir.

2.1.4 Matplotlib

Matplotlib, kullanıcıların farklı türlerde grafikler oluşturmasını sağlayan geniş bir işlevseliteye sahiptir. Çizgi grafikleri, sütun grafikleri, pasta grafikleri ve dağılım grafikleri gibi birçok farklı grafik türünü destekler. Bu sayede projede, tekstil ürünlerinin kusurlarını göstermek için çeşitli grafikler oluşturulabilir.

Matplotlib'in basit ve anlaşılır bir arayüzü vardır. Verileri temsil etmek ve görsel olarak sunmak için kullanıcı dostu bir yol sunar. Veri tabanlı analizlerin sonuçlarını anlamak ve paylaşmak için grafiklerin oluşturulması için tercih edilen bir araçtır.

Projede, Matplotlib kütüphanesi kullanılarak, tekstil ürünlerinin kusurlarını gösteren grafikler oluşturulabilir. Örneğin, bir sütun grafik üzerinde farklı kusur türlerinin dağılımı veya bir çizgi grafik üzerinde zaman içindeki kusur sayılarının değişimi gibi bilgiler görselleştirilebilir.

2.2 .NET Framework

.Net (dotNet), Microsoft tarafından geliştirilen bir yazılım platformudur. Bu platform, Windows işletim sistemi üzerinde uygulama geliştirmeyi kolaylaştırmak, verimliliği

artırmak ve genel olarak yazılım geliştirme sürecini iyileştirmek amacıyla tasarlanmıştır.

.Net, çok çeşitli programlama dilleriyle uyumlu bir şekilde çalışabilen bir çerçeve sunar. Bu diller arasında C#, Visual Basic.NET (VB.NET), F# ve daha birçok dil bulunur. Bu, geliştiricilere seçtikleri dilde çalışma özgürlüğü sağlar ve platformun esnekliğini artırır.

.Net'in temel amacı, güvenilir, verimli ve ölçeklenebilir uygulamaların geliştirilmesini desteklemektir. Platform, birçok hizmeti bir araya getirir ve uygulamaların daha kolay ve daha hızlı bir şekilde oluşturulmasını sağlar. Bu hizmetler arasında derleme (compilation), hata ayıklama (debugging), bellek yönetimi (memory management), güvenlik (security) ve veritabanı erişimi (database access) gibi birçok önemli bileşen yer alır.

.Net, yazılım geliştirme sürecini hızlandıran birçok araç ve kütüphane sunar. Bunlar arasında Visual Studio gibi geliştirme ortamları, .Net Framework ve .Net Core gibi çerçeveler, ASP.NET gibi web uygulama geliştirme araçları ve Entity Framework gibi veritabanı erişim araçları bulunur. Bu araçlar ve kütüphaneler, geliştiricilere güçlü bir altyapı sağlar ve kod tekrarını azaltarak daha verimli bir şekilde çalışmalarını sağlar.

.Net'in bir diğer önemli özelliği, platformunun taşınabilirliğidir. Microsoft, .Net Core adı verilen açık kaynaklı bir versiyonunu geliştirmiştir. .Net Core, Windows, macOS ve Linux gibi farklı işletim sistemlerinde çalışabilir ve bu da uygulamaların farklı platformlara taşınabilmesini sağlar.

2.3 C#

C#, Microsoft tarafından geliştirilen, çok yönlü bir programlama dilidir. C#, basit, modern ve nesne odaklı bir yapıya sahiptir ve genellikle Windows tabanlı uygulamaların geliştirilmesinde kullanılır. Dilin temel amacı, geliştiricilere güvenli, verimli ve ölçeklenebilir uygulamalar oluşturma imkanı sunmaktır.

C# dilinin tasarımı, C++ ve Java gibi dillerden etkilenmiştir. Dilin sentaksı, C++'ın güçlü ve esnek yapısını korurken, Java'nın platform bağımsızlığını ve güvenlik özelliklerini benimsemiştir. Bu sayede C#, kullanıcı dostu bir dil olarak kabul edilir ve geliştiricilere daha kolay bir öğrenme süreci sunar.

C#'ın nesne odaklı yapısı, yazılım projelerini modüler bir şekilde organize etmeyi sağlar. Sınıflar, nesneler ve kalıtım gibi nesne odaklı programlamanın temel bileşenleri C# dilinde mevcuttur. Bu sayede geliştiriciler, kodlarını daha anlaşılır, bakımı kolay ve yeniden kullanılabilir hale getirebilirler.

C# dilinin bir diğer önemli özelliği, geniş bir .NET Framework kütüphane desteğine sahip olmasıdır. .NET Framework, geliştiricilere birçok hazır bileşen, araç ve hizmet sunar. Bu kütüphane, grafik işleme, veritabanı erişimi, web uygulamaları geliştirme, ağ programlama gibi çeşitli alanlarda işlevselliği artırmak için kullanılabilir. Ayrıca, C# dilinin zengin ve kapsamlı bir standart kütüphanesi bulunur, bu da geliştirme sürecini hızlandırır ve tekrar kullanılabilir kod parçaları sağlar.

C# dilinin kullanım alanları oldukça geniştir. Windows masaüstü uygulamaları, oyun geliştirme, web uygulamaları, veritabanı uygulamaları, mobil uygulamalar gibi çeşitli projeler C# dilinde geliştirilebilir. C# ayrıca, ASP.NET gibi web geliştirme araçlarıyla birlikte kullanılarak dinamik ve etkileşimli web siteleri ve hizmetleri oluşturmak için de tercih edilir.

2.4 Google Colab

Google Colab (Colaboratory), Google tarafından sunulan bir bulut tabanlı Jupyter Notebook hizmetidir. Jupyter Notebook, interaktif programlama, veri analizi ve makine öğrenimi gibi işlemleri gerçekleştirmek için popüler bir araçtır. Google Colab ise Jupyter Notebook'u bulut tabanlı bir ortamda çalıştıran ücretsiz bir hizmet olarak öne çıkar.

Google Colab, kullanıcılara Python kodunu tarayıcı üzerinden çalıştırma ve paylaşma imkanı sunar. Bu sayede herhangi bir kurulum yapmadan, kullanıcılar sadece internete erişerek Python kodlarını yazabilir ve çalıştırabilirler. Ayrıca, Colab ile oluşturulan not defterleri, Google Drive'a kaydedilebilir ve paylaşılabilir.

Colab, kullanıcılara GPU (Graphics Processing Unit) ve TPU (Tensor Processing Unit) gibi yüksek performanslı donanım kaynaklarını ücretsiz olarak kullanma imkanı sağlar. Bu özellik, derin öğrenme ve büyük veri işleme gibi hesaplama yoğun işlemleri hızlandırmak için önemli bir avantaj sunar.

Google Colab ayrıca, birçok popüler Python kütüphanesini önceden yüklenmiş olarak sunar. Bu sayede kullanıcılar, projelerinde yaygın olarak kullanılan kütüphaneleri hızlı

bir şekilde kullanabilirler. Ayrıca, Colab'da hazır bulunan örnek projeler ve eğitimler de kullanıcıların başlangıç yapmasına yardımcı olur.

Özetlemek gerekirse, Google Colab, Jupyter Notebook'u bulut tabanlı bir ortamda çalıştıran ücretsiz bir hizmettir. Kullanıcılar, tarayıcı üzerinden Python kodlarını yazabilir, çalıştırabilir ve paylaşabilirler. Ücretsiz GPU/TPU erişimi ve önceden yüklenmiş kütüphaneler gibi avantajlarıyla veri analizi, makine öğrenimi ve derin öğrenme gibi projeler için kullanıcı dostu bir çözüm sunar.

Bu sebeplerden ötürü Görüntü İşleme Modelinin eğitimi için Google Colab tercih edildi.

3. METODOLOJİ

3.1 Proje Geliştirme Süreci

Çalışma planı hazırlamak projenin gidişatı, projedeki adımların görülmesi, çıkabilecek hataların tespiti gibi unsurlarda önemlidir. Projemizin başında belirlediğimiz zaman akışı aşağıdaki gibidir.

- Araştırma ve Veri Toplama (1 hafta): İlgili makaleleri araştırma ve veri kümesi oluşturma süreci için gerekli verilerin toplanması
- Derin Öğrenme Algoritmalarının Eğitimi (2 hafta): TensorFlow kullanarak derin öğrenme algoritmalarının eğitimi. Bu süreçte, algoritmaların parametreleri ayarlanacak ve doğruluk oranı arttırılmaya çalışılacak.
- Görüntü İşleme Tekniklerinin Öğrenilmesi (1 hafta): Görüntü işleme teknikleri öğrenilerek, kumaş hatalarının tespiti için kullanılacak yöntemler incelenecek ve seçilecek.
- Derin Öğrenme Algoritmalarının Uygulanması (2 hafta): Eğitilen algoritmaların kullanarak, kumaş hatalarının tespiti yapılacak. Bu süreçte, görüntü işleme teknikleri kullanılacak ve algoritmaların daha doğru sonuçlar üretmesi sağlanacak.

- .NET Framework ve Arayüz Tasarımı (3 hafta): .NET Framework kullanarak, arayüz tasarlanacak. Bu süreçte, .NET Framework öğrenilerek, arayüz için gerekli bileşenlerin kullanımı öğrenilecek.
- Entegrasyon ve Test Süreci (1 hafta): Derin öğrenme algoritmaları, görüntü işleme teknikleri ve arayüzün bir araya getirilerek, projenin entegrasyonu yapılacaktır. Ardından testler gerçekleştirilecek.

3.2 Proje Taslağının Oluşturulması

Projenin ilk aşamasında bilgisayarda bulunan kamera saptanması için bir windows for uygulaması penceresinde kamera yakalanımı sağlanır. Açılan kamera görüntüsü fotoğraf olacak şekilde çekilir. Daha sonra isim yazılarak kaydedilen fotoğraf bir dosya yoluna kaydedilir. Kaydedilen dosya konumunda bulunan fotoğraf verisi http isteği ile python API'sine gönderilir. Çekilen veri üzerinde kumaş hatasının tespiti yapılır ve yakalanan hata etiketlenerek gösterilir. İşlemlerin sonucunda python HTTP isteğine dönüş olarak işlenmiş görüntüyü döndürür.

3.3 Araçların Kurulumu ve Yönetimi

Bu projenin gerçekleştirilmesi için bazı önemli araçların kurulumu ve yönetimi gerekmektedir. İlk olarak, PyTorch gibi bir makine öğrenmesi kütüphanesinin kurulumu önemlidir. PyTorch, derin öğrenme modelleri oluşturmak, eğitmek ve değerlendirmek için kullanılan popüler bir kütüphanedir. PyTorch'un resmi web sitesinden indirme bağlantısı bulunmaktadır. Projenin odak noktası olan kumaş hatalarının tespiti için derin öğrenme modelleri kullanmayı hedefliyoruz ve PyTorch bu süreçte bize birçok avantaj sağlamaktadır.

PyTorch'un avantajlarından biri, kolay kullanılabilir bir API sunmasıdır. İntuitif bir arayüz ve kullanıcı dostu bir yapıya sahip olması, model oluşturma ve eğitim sürecini hızlandırırken geliştirme sürecini de kolaylaştırmaktadır. Ayrıca PyTorch, Python programlama dilini temel alır, bu da Python ekosistemi ile uyumlu çalışabilme imkanı sağlar ve geniş bir kullanıcı topluluğuyla desteklenir.

PyTorch'un bir diğer önemli avantajı, dinamik graf hesaplamaları yapabilme yeteneğidir. Bu, modelin yapısını eğitim süreci boyunca dinamik olarak değiştirebilme imkanı verir. Graf yapısının dinamik olarak ayarlanabilmesi, modelin esnekliğini

artırır ve daha karmaşık modellerin oluşturulmasını kolaylaştırır. Ayrıca, hata ayıklama ve modelin geliştirilmesi sırasında graf yapısının daha kolay anlaşılmasını sağlar.

PyTorch ayrıca GPU hızlandırmasını destekler. Derin öğrenme modelleri genellikle büyük miktarda veri ve hesaplama gücü gerektirir. PyTorch, CUDA üzerinden GPU hızlandırmasını kullanarak model eğitimi ve tahminleri hızlandırabilir. Bu, işlem sürelerini büyük ölçüde azaltır ve daha büyük ve karmaşık veri setlerini işleyebilme yeteneği sağlar.

Sonuç olarak, PyTorch'un bu projede kullanımı, kullanıcı dostu arayüzü, dinamik graf hesaplamaları, Python uyumluluğu ve GPU hızlandırma desteği gibi avantajlar sunar. Bu avantajlar, derin öğrenme modelinin geliştirilmesi ve kumaş hatalarının tespit sürecinin daha verimli, esnek ve hızlı bir şekilde gerçekleştirilmesini sağlar.

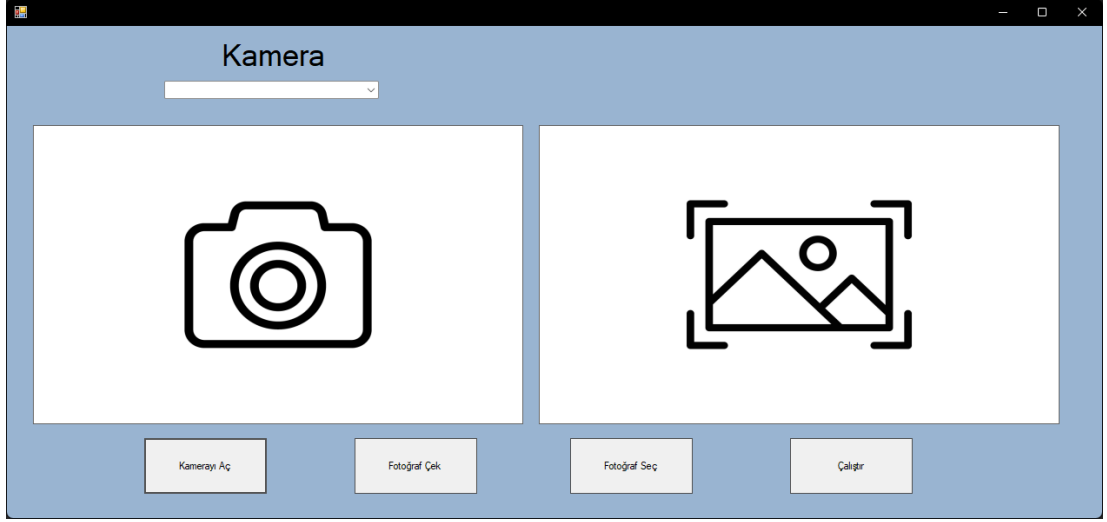
Projenin geliştirme sürecinde, bir entegre geliştirme ortamı (IDE) olarak Visual Studio ve Visual Studio Code kullanılmıştır. Visual Studio, geniş kapsamlı bir IDE olup, C# ve .NET gibi Microsoft teknolojileriyle uyumludur. Proje sürecinde kamera için tasarlanan arayüzde C# programlama dili kullanılmaktadır. Visual Studio Code ise hafif bir metin düzenleyici ve geliştirme ortamıdır. Python ve diğer dillerle çalışmak için de kullanılabilir. Bu geliştirme ortamları sayesinde daha kolay bir şekilde iki dil entegre edilebilmektedir.

Bu araçların kurulumu ve yönetimi, projenin başarılı bir şekilde tamamlanması ve hedeflenen sonuçların elde edilmesi için önemlidir. Doğru araçların seçimi, doğru şekilde kurulması ve etkin bir şekilde yönetilmesi, projenin geliştirme sürecini kolaylaştırır, hataları azaltır ve verimliliği artırmaktadır.

3.4 Kamera Arayüzü

Bilgisayar kamerası kullanılarak gerekli kumaşın hatası varsa tespiti için gerekli .Net frameworku kullanılarak C# diliyle yazılmış windows form uygulaması arayüzüdür.

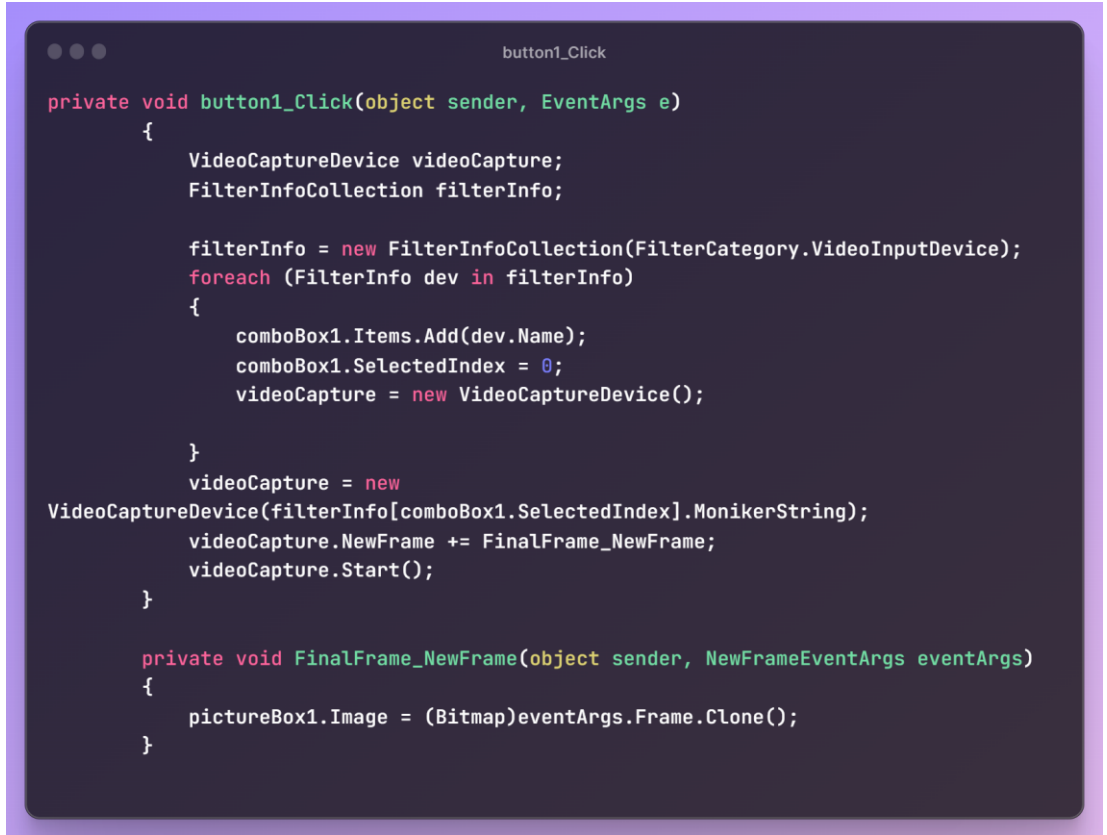
Bu arayüz sayesinde kamera kullanımı sağlanır, açılan kamera ile kumaş görüntüsü alınarak gerekli hatanın tespiti için derin öğrenme modeline iletimi sağlanmaktadır.



Şekil 3.1 : Kamera Arayüz Tasarımı

Şekil 3.1’de C# dilinde windows form uygulaması şeklinde açtığımız projemizin kamera arayüz tasarımı bulunmaktadır.

Soldaki kutu kamera görüntüsünü vermektedir. Sağdaki kutuda ise kamerayı kullanarak çektiğimiz ya da dahili hafızadan yüklediğimiz görsel gözükmektedir.



Şekil 3.2 : Kamera Başlangıç Butonu Fonksiyonu

Bu kod parçası, “Kamerayı Aç” düğmesine tıklanıldığında gerçekleşecek olan olayı tanımlar. Olay, `button1_Click` adında bir olay işleyicisi (event handler) tarafından işlenir. “`VideoCaptureDevice`” ve “`FilterInfoCollection`” türünden iki değişken tanımlanır: “`videoCapture`” ve “`filterInfo`”. “`filterInfo`” değişkenine, “`FilterCategory.VideoInputDevice`” kategorisine ait video giriş cihazlarının bilgilerini içeren bir koleksiyon atanır. “`foreach`” döngüsü kullanılarak, her bir cihazın bilgisi “`filterInfo`” koleksiyonundan alınır ve adı “`comboBox1`” adlı bir `ComboBox` kontrolüne eklenir. “`comboBox1.SelectedIndex`” özelliği 0 olarak ayarlanarak, `ComboBox`'ın varsayılan olarak ilk öğeyi seçmesi sağlanır. “`videoCapture`” değişkenine yeni bir “`VideoCaptureDevice`” örneği oluşturulur. “`videoCaptur`”e değişkenine, “`filterInfo`” koleksiyonundan seçilen cihazın “`MonikerString`” özelliği atanır. Bu, seçilen video cihazını temsil eden bir kimlik dizisidir. “`videoCapture`” değişkeninin “`NewFrame`” olayına “`FinalFrame_NewFrame`” adlı bir olay işleyici atanır. Bu işleyici, yeni bir görüntü çerçevesi yakalandığında tetiklenecektir. “`videoCapture.Start()`” yöntemi kullanılarak, video yakalama işlemi başlatılır ve cihazdan görüntüler alınmaya başlar. Bu kod parçası, bir video cihazı seçimi yapılmasını ve seçilen cihazdan görüntü almayı sağlar.

Tasarımın 2. bloğunda ise kameradan yakalanacak ve arka planda işlenecek fotoğrafın çekileceği kısım bulunmaktadır. “Fotoğraf Çek” butonuna basılarak fotoğraf çekilip kaydedilmektedir.


```
button2_Click

private void button2_Click(object sender, EventArgs e)
{
    pictureBox3.Image = pictureBox1.Image;
    //filename location

    imageUrl = @"C:\Users\Public\Pictures\kaydedilen.jpg";

    var bitmap = new Bitmap(pictureBox3.Width, pictureBox3.Height);
    pictureBox3.DrawToBitmap(bitmap, pictureBox3.ClientRectangle);
    System.Drawing.Imaging.ImageFormat imageFormat = null;
    imageFormat = System.Drawing.Imaging.ImageFormat.Jpeg;

    // save the image

    bitmap.Save(imageLocation);
}
```

Şekil 3.3 : Fotoğraf Yakalama Butonu Fonksiyonu

Şekil 3.4’te , tasarımda bulunan “Capture” düğmesine tıklanıldığında gerçekleşecek olan olayı tanımlar. Olay, button2_Click adında bir olay işleyicisi (event handler) tarafından işlenir. “pictureBox1” kontrolündeki görüntüyü “pictureBox3” kontrolüne kopyalar. Bu, “pictureBox1.Image” özelliğinin “pictureBox3.Image” özelliğine atanmasıyla gerçekleştirilir. Yeni bir “Bitmap” nesnesi oluşturulur ve bu nesneye “pictureBox3” kontrolünün genişlik ve yükseklik değerleri verilir. “pictureBox3” kontrolünün içeriğini, oluşturulan “Bitmap” nesnesine çizdirilir. Bu, “pictureBox3.DrawToBitmap” yöntemi kullanılarak gerçekleştirilir. “imageFormat” adlı bir “ImageFormat” nesnesi tanımlanır ve başlangıçta null olarak atanır. Görüntünün hangi dosya biçiminde kaydedileceğini belirlemek için “imageFormat” değişkeni, “ImageFormat.Jpeg” değeriyle yeniden atanır. Oluşturulan “Bitmap” nesnesi, belirtilen dosya adı ve biçimi kullanılarak kaydedilir. Bu, “bitmap.Save” yöntemiyle gerçekleştirilir.

Bu kod parçası, pictureBox1 kontrolünde görüntülenen resmi pictureBox3 kontrolüne kopyalayarak ve kullanıcı tarafından belirtilen bir dosya adı ve konumunda kaydederek görüntüyü kaydetme işlemini gerçekleştirir.

“Fotoğraf yükle” butonuna basarak açılan pencerede istenilen fotoğrafa erişilebilmekte ve arayüzde görülebilmektedir.

```
button3_Click

public void button3_Click(object sender, EventArgs e)
{
    try
    {
        OpenFileDialog dialog = new OpenFileDialog();
        dialog.Filter = "jpg files (*.jpg)|*.jpg| PNG files (*.png)|*.png| All Files (*.*)|*.*";

        if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            imageLocation = dialog.FileName;
            pictureBox3.ImageLocation = imageLocation;
        }
    }
    catch (Exception)
    {
        MessageBox.Show("An Error Occured", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Şekil 3.4 : Fotoğraf Yükleme Butonu Fonksiyonu

Şekil 3.4’te, “Fotoğraf Yükle” düğmesine tıklanıldığında gerçekleşecek olan olayı tanımlar. Olay, button3_Click adında bir olay işleyicisi (event handler) tarafından işlenir. “imageLocation” adında bir boş bir dize tanımlanır. Bu değişken, seçilen görüntünün dosya konumunu tutmaktadır. “OpenFileDialog” sınıfından bir nesne oluşturulur. Bu nesne, kullanıcının bir dosya seçmesine yardımcı olur. Filter özelliği kullanılarak, kullanıcının sadece belirli türdeki dosyaları seçmesine izin verilir. Örneğin, sadece “JPG” veya “PNG” dosyalarını seçmesine izin verilebilir. “ShowDialog” yöntemi çağrılarak, dosya seçme iletişim kutusu görüntülenir ve kullanıcının bir dosya seçmesi beklenir. Eğer kullanıcı bir dosya seçmişse “(DialogResult.OK)”, seçilen dosyanın konumu “imageLocation” değişkenine atanır. “pictureBox3” kontrolünün “ImageLocation” özelliği, “imageLocation” değişkeninin değeriyle güncellenir. Bu, seçilen görüntünün “pictureBox3” kontrolünde görüntülenmesini sağlar. Eğer bir hata oluşursa, bir hata iletişim kutusu görüntülenir ve kullanıcıya bir hata mesajı gösterilir.

Bu kod parçası, kullanıcının bir dosya seçmesine olanak sağlayan bir iletişim kutusu görüntüler ve seçilen dosyanın konumunu alarak “pictureBox3” kontrolünde görüntülenmesini sağlar.

```
button3_Click

public void button4_Click(object sender, EventArgs e)
{

    string message = "";
    string url = "http://localhost:5000/api/path"; // İstek gönderilecek URL

    string postData = imageLocation;
    // İstek oluşturma
    HttpRequest request = (HttpRequest)WebRequest.Create(url);
    request.Method = "POST";
    request.ContentType = "application/json";

    // Veriyi isteğe ekleyerek gönderme
    using (StreamWriter writer = new StreamWriter(request.GetRequestStream()))
    {
        writer.Write(postData);
    }
    string responseText = "";
    try
    {
        // İstek gönderme ve yanıt alma
        HttpResponse response = (HttpResponse)request.GetResponse();

        // Yanıtı okuma
        using (StreamReader reader = new StreamReader(response.GetResponseStream()))
        {
            responseText = reader.ReadToEnd();
            Console.WriteLine(responseText);
            message = "Başarılı";
            MessageBox.Show(message);
        }
    }
    catch (WebException ex)
    {
        // İstek hatası durumunda hata mesajını yakalama
        Console.WriteLine(ex.Message);
        message = "Bir Sorunla Karşılaşıldı";
        MessageBox.Show(message);
    }

    pictureBox3.ImageLocation = $"C:\\Users\\Kullanıcı\\Desktop\\Yolov7\\{responseText}";
}
```

Şekil 3.5 : 'Çalıştır' Butonu fonksiyonu

Şekil 3.5'te Çalıştır butonuna tıklandığında çalışacak fonksiyon gözükmemektedir. Bu fonksiyon kamera kullanılarak ya da fotoğraf yükleme özelliği ile yüklenilmiş olan 'PictureBox3'te mevcut olan fotoğrafın dosya konumunu POST yöntemi ile Python API'mizin çalıştığı Localhost:5000'e gönderir.

Verinin isteğe eklenerek gönderilmesi için StreamWriter kullanılır. StreamWriter ile veri, isteğin gövdesine yazılır.

Sonraki adımda, isteğin gönderilmesi ve yanıtın alınması gerçekleştirilir. request.GetResponse() metodu ile istek gönderilir ve yanıt alınır. Yanıt okunur ve responseText değişkenine aktarılır. Ayrıca responseText konsola yazdırılır. message değişkenine "Başarılı" değeri atanır ve bir mesaj kutusuyla "Başarılı" mesajı gösterilir.

Eğer bir hata oluşursa, WebException türündeki hatalar yakalanır. Hata mesajı konsola yazdırılır, message değişkenine "Bir Sorunla Karşılaşıldı" atanır ve bir mesaj kutusuyla "Bir Sorunla Karşılaşıldı" mesajı gösterilir.

Son olarak, pictureBox3 kontrolüne görüntünün yolunu atamak için pictureBox3.ImageLocation özelliği kullanılır. Yol, responseText değişkeni ve belirli bir dizinle birleştirilerek oluşturulur.

Bu kod, imageLocation değişkenindeki bir görüntüyü belirtilen URL'ye POST isteği yaparak gönderir ve yanıt olarak alınan bir görselin yolunu pictureBox3 kontrolüne atar. Ayrıca, isteğin başarılı veya hatalı olup olmadığına bağlı olarak mesajlar gösterir.

3.5 Verilerin Toplanması

Başarılı bir görüntü işleme modeli eğitmek için büyük oranda eğitim verisine ihtiyaç vardır. Bu ihtiyacı karşılamak için bizim ilk tercihimiz kaggle.com oldu. Kaggle, veri bilimcilerin ve makine öğrenimi mühendislerinin projelerini geliştirmek, veri setleriyle çalışmak ve yarışmalara katılmak için kullanabilecekleri bir platformdur. Kaggle, açık bir topluluk tabanlı veri bilimi platformudur ve kullanıcılarına veri setlerini keşfetme, analiz etme, görselleştirme, model oluşturma ve paylaşma imkanı sunar.

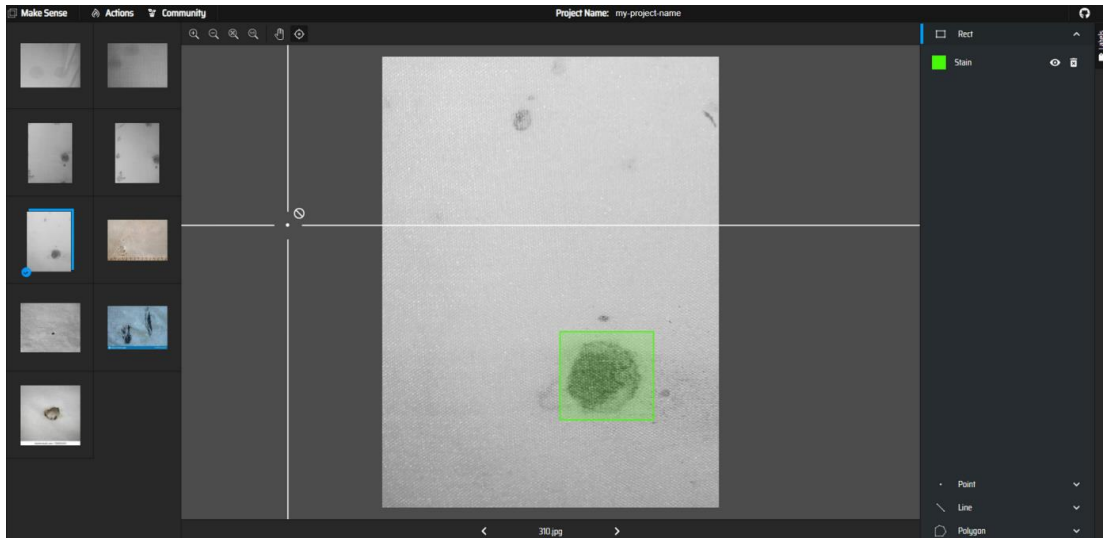
Kaggle'da kullanıcılar, çeşitli veri setlerini indirebilir, veri analizi ve keşfi yapabilir, makine öğrenimi modelleri oluşturabilir ve bu modelleri yarışmalara veya projelere uygulayabilirler. Ayrıca, kullanıcılar Kaggle topluluğunda diğer veri bilimcilerle iletişim kurabilir, fikir alışverişinde bulunabilir ve bilgi paylaşabilirler.

Sadece Kaggle üzerinde bulduğumuz verilerin yeterli olmadığını düşündük ve Stok fotoğraf hizmeti veren siteler ve google görsel arama üzerinden bulabildiğimiz kumaş fotoğrafları ile veri setimizi destekledik.

3.6 Verilerin Sınıflandırılması

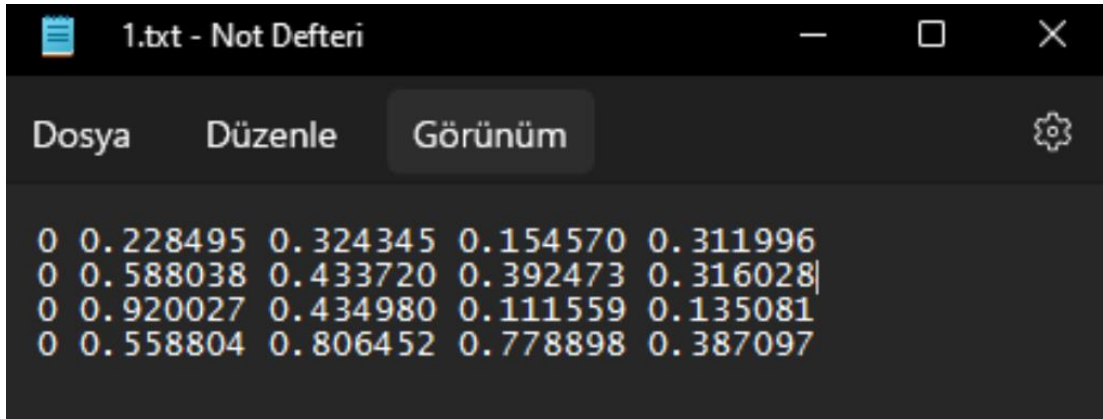
Topladığımız verileri eğitmek üzere modele vermeden önce verilerin etiketlenmesi gerekir. Bu işlem için ise makesense.ai websitesini kullandık.

Makesense.ai, görüntü işleme ve etiketleme için bir yapay zeka platformudur. Bu platform, veri setlerini etiketlemek ve eğitim verileri oluşturmak için kullanılan bir araç sağlar. Görüntü işleme projelerinde, özellikle nesne tanıma ve görüntü sınıflandırma gibi alanlarda, doğru etiketlemeye ihtiyaç duyulur.



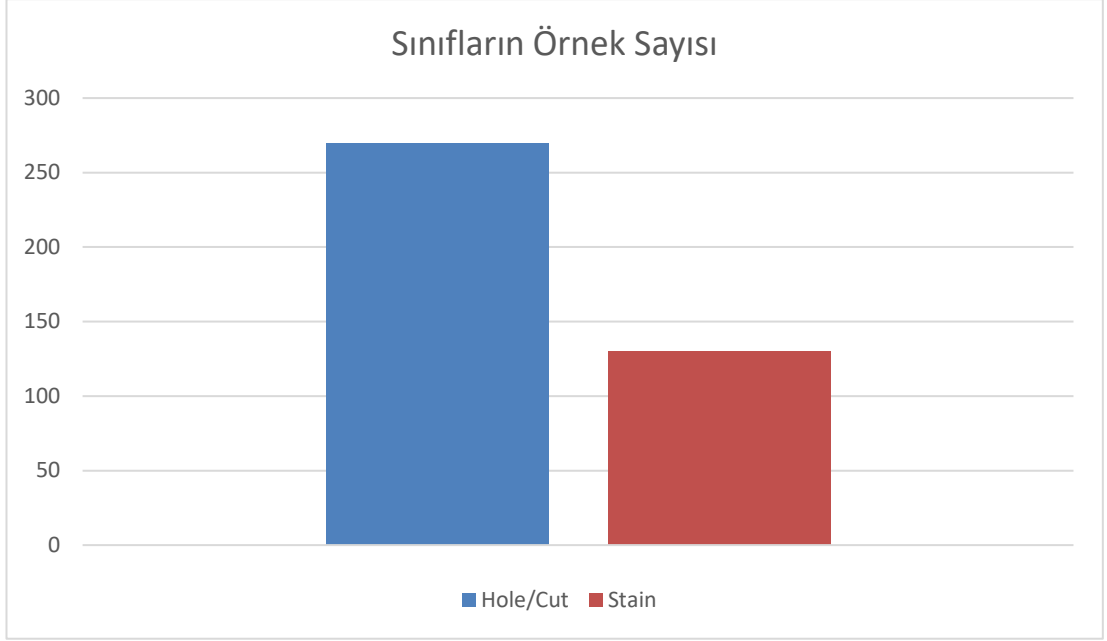
Şekil 3.6 : Makesense.ai üzerinde veri etiketleme

Verilerin etiketlenmesi bittikten sonra her görselle aynı isimde bir .txt dosyası indirilir.



Şekil 3.7 : Etiketlenmiş bir görselin sınıf ve koordinatlarını içeren .txt dosyası

Yukarıdaki görselde: '0' nesnenin sınıfıdır. Koordinatlar ise işaretlemeye kullanılan dörtgenin her bir köşesinin koordinatlarıdır. Her yeni satır görsel üzerindeki etiketli bir başka nesneyi işaret eder.



3.7 Modelin Eğitimi

Model eğitimi için YOLOv7 Nesne algılama modelinden faydalandık. OLOv7, PyTorch kütüphanesi ile yazılmıştır. YOLOv7'nin kaynak kodu, GitHub'da mevcuttur ve açık kaynaklı bir yazılımdır. Nesne tespiti için derin öğrenme yöntemlerinden biri olan Convolutional Neural Network (CNN) algoritmalarını kullanır.

YOLOv7'nin birden fazla Modeli bulunuyor. Aşağıdaki Tabloda Farklı Modellerin Performanslarını görebilirsiniz. Projemizde kullandığımız nihai model en başarılı sonucu aldığımız YOLOv7 oldu.

Model	Test Size	AP ^{test}	AP ₅₀ ^{test}	AP ₇₅ ^{test}	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 fps	2.8 ms
YOLOv7-X	640	53.1%	71.2%	57.8%	114 fps	4.3 ms
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 fps	7.6 ms
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 fps	12.3 ms
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 fps	15.0 ms
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 fps	18.7 ms

Şekil 3.8 : Modeller ver Performans Metrikleri

Bu modeli kendi ihtiyaçlarımız doğrultusundan değiştirerek projemize uygun bir hale getirdik.


```
Custom.yaml

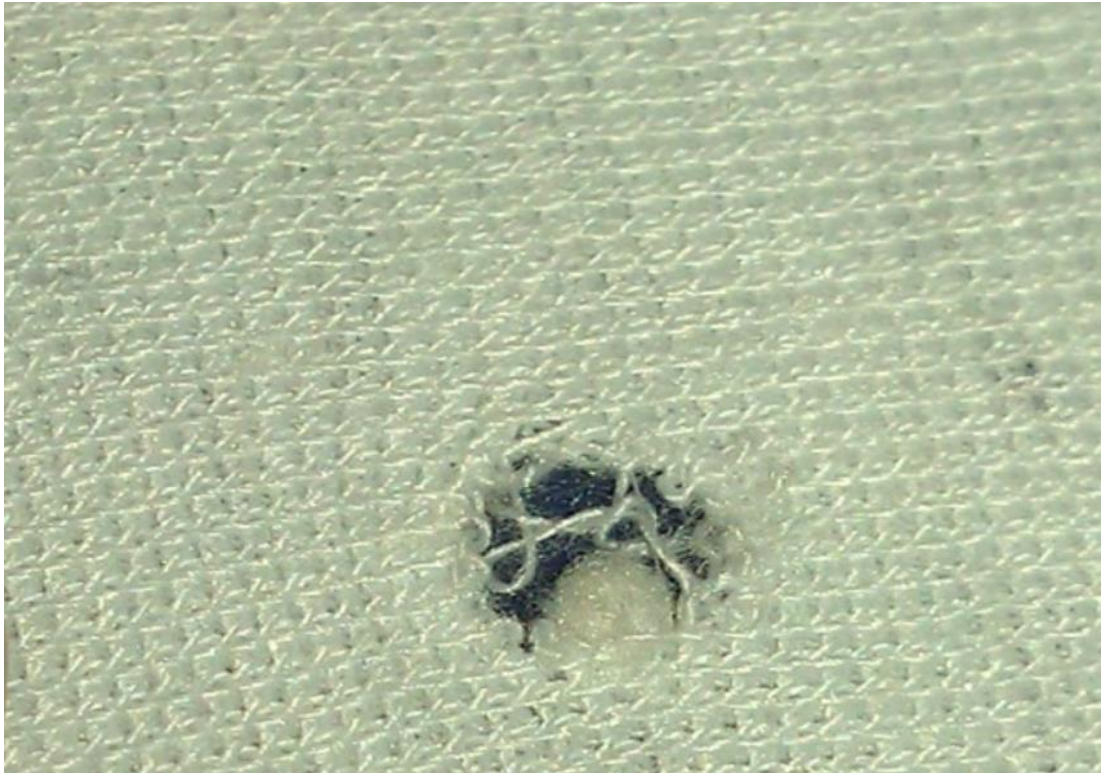
train: ./data/train
val:  ./data/val

# number of classes
nc: 2

# class names
names: [ 'stain', 'hole/cut' ]
```

Şekil 3.9 : Sınıfların belirtildiği Custom.yaml dosyası

Modelin kendisinde 80 adet sınıf bulunmakta fakat biz kendi .yaml formatındaki dosyamızı oluşturup 2 sınıf belirtiyoruz. Bu sınıflar lekeli kumaşlar için ‘stain’ ve delik veya kesik kumaşlar için ‘hole/cut’.



Şekil 3.10 : 'Hole/Cut' sınıfı örnek görsel



Şekil 3.11 : 'Stain' sınıfı örnek görsel

```
!git clone https://github.com/WongKinYiu/yolov7.git

!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt

!python train.py --device 0 --batch-size 16 --epochs 100 --img 640 640 --data
data/custom_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg cfg/training/yolov7.yaml --
weights yolov7.pt --name yolov7-custom
```

Şekil 3.12 : Yolov7'yi GitHub'tan klonlamak ve Model eğitimi

Yukarıdaki kod, YOLOv7 isimli bir nesne algılama modelini GitHub üzerinden klonlamak ve eğitmek için kullanılan komutları içeriyor. Aşağıda, her bir komutun ne yaptığını açıklanmakta:

!git clone https://github.com/WongKinYiu/yolov7.git:

Bu komut, belirtilen GitHub adresindeki YOLOv7 modelini yerel bir klasöre kopyalamak için Git'in klonlama işlemini gerçekleştirir. Böylece, modelin tüm dosyalarına erişebilirsiniz.

!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt:

Bu komut, önceden eğitilmiş bir YOLOv7 modelinin ağırlıklarını indirmek için Wget aracını kullanır. İndirilen yolov7.pt dosyası, eğitimde kullanılacak başlangıç noktası olarak kullanılacaktır.

```
!python train.py --device 0 --batch-size 16 --epochs 100 --img 640 640 --data
data/custom_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg
cfg/training/yolov7.yaml --weights yolov7.pt --name yolov7-custom:
```

Bu komut, YOLOv7 modelini eğitmek için Python betiğini çalıştırır. Aşağıda parametrelerin açıklamalarını bulabilirsiniz:

--device 0: Eğitimin GPU üzerinde yapılmasını sağlar (0, birinci GPU'yu temsil eder).

--batch-size 16: Her bir eğitim adımında kullanılacak örnek sayısını belirler.

--epochs 100: Eğitimin kaç epoch (iterasyon) süreceğini belirler.

--img 640 640: Eğitimde kullanılacak görüntülerin boyutunu piksel cinsinden belirler.

--data data/custom_data.yaml: Eğitim veri setinin konfigürasyon dosyasını belirtir. Bu dosya, veri seti yolu, sınıf etiketleri vb. bilgileri içerir.

--hyp data/hyp.scratch.custom.yaml: Eğitim hiperparametrelerinin konfigürasyon dosyasını belirtir.

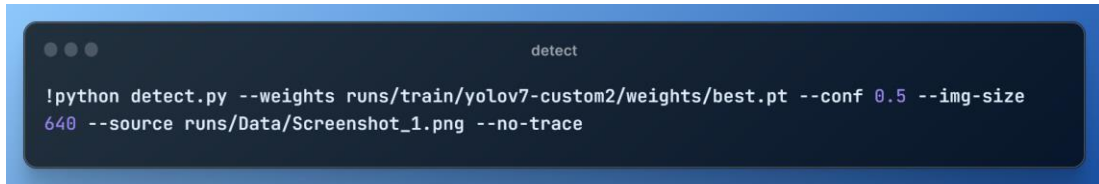
--cfg cfg/training/yolov7.yaml: Modelin konfigürasyon dosyasını belirtir.

--weights yolov7.pt: Başlangıç noktası olarak kullanılacak önceden eğitilmiş ağırlıkların yolunu belirtir.

--name yolov7-custom: Eğitim sürecinde oluşacak çıktılar için bir isim belirtir. Bu isim, kaydedilen model ve günlük dosyalarının adında kullanılır.

Eğitimin tamamlanmasının ardından 100 iterasyon sonucunda en iyi sonucu veren ağırlıklar best.pt adıyla kaydedilir. Bu dosya sınıflandırmada kullanacağımız ağırlıkları içerir.

Modelimizi test etmek amacıyla Google Colab üzerinde deneme yapıyoruz.



```
detect
!python detect.py --weights runs/train/yolov7-custom2/weights/best.pt --conf 0.5 --img-size 640 --source runs/Data/Screenshot_1.png --no-trace
```

Şekil 3.13 : Detect.py dosyasına gerekli parametreleri göndermek ve çalıştırmak

`!python detect.py`: Bu komut, `detect.py` adlı Python betiğini çalıştırır ve nesne tespiti işlemini başlatır.

`--weights runs/train/yolov7-custom2/weights/best.pt`: Bu parametre, kullanılacak YOLOv7 modelinin ağırlıklarının yolunu belirtir. `runs/train/yolov7-custom2/weights/best.pt` ifadesi, eğitim sonucunda elde edilen en iyi modelin ağırlıklarını gösterir.

`--conf 0.5`: Bu parametre, tespit edilen nesnelerin minimum güven değerini belirtir. 0.5 değeri, nesnelerin en az %50 güvenlikle tespit edilmesi gerektiğini ifade eder.

`--img-size 640`: Bu parametre, giriş görüntülerinin boyutunu piksel cinsinden belirtir. Bu durumda, 640x640 boyutunda görüntüler kullanılır.

`--source runs/Data/Screenshot_1.png`: Bu parametre, tespit yapılacak görüntünün yolunu belirtir. `runs/Data/Screenshot_1.png` ifadesi, tespit yapılacak görüntünün dosya yolunu gösterir. Bu kodda, `Screenshot_1.png` adlı bir görüntü kullanılır.

`--no-trace`: Bu parametre, nesne tespiti sonuçlarının çizimleme izlerinin kaydedilmesini engeller. Yani, çıktı görüntüsünde nesnelerin çevrelerinin izi olmayacaktır.



Şekil 3.14 : Test Görseli, model kumaştaki deliği tespit etti



Şekil 3.15 : Test görseli, Model Kumaş üzerindeki lekeyi tespit etti

Modelimizi test ettiğimizde yırtık veya delikleri yüksek bir güven oranıyla tespit ederken leke tespitinde tespit etmiş olsa da ‘hole/cut’ sınıfı kadar başarılı bir şekilde

tanımlayamadığını görebiliyoruz. Daha başarılı sonuçlar için veri kümemizdeki ‘stain’ sınıfı örneklerini arttırabiliriz.

3.8 Modelin performansı

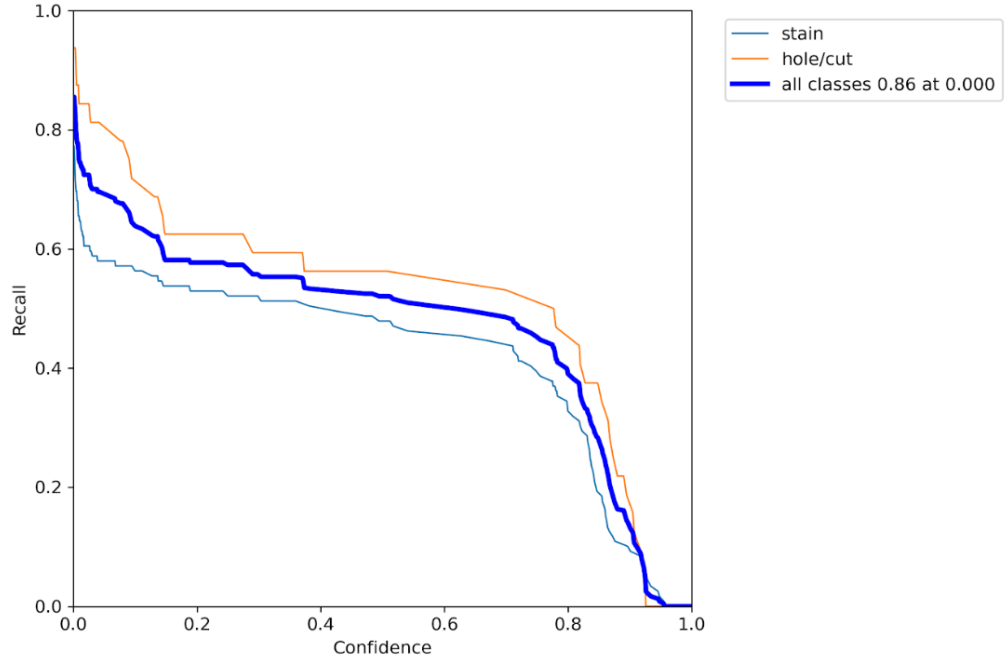


Şekil 3.16 : Precision ve Recall Metrikleri. X eksenİ İterasyon sayısıdır.

Recall (Hatırlama): Hatırlama, gerçek pozitiflerin (true positives) tespit edilme oranını ifade eder. Yani, gerçekte pozitif olan örneklerin, model tarafından ne kadarının doğru bir şekilde pozitif olarak sınıflandırıldığını gösterir. Hatırlama, yanlış negatiflerin (false negatives) oluştuğu durumlarda düşük olabilir ve eksik sonuçları ifade eder. Yüksek hatırlama değeri, modelin pozitif örnekleri daha iyi yakaladığını gösterir.

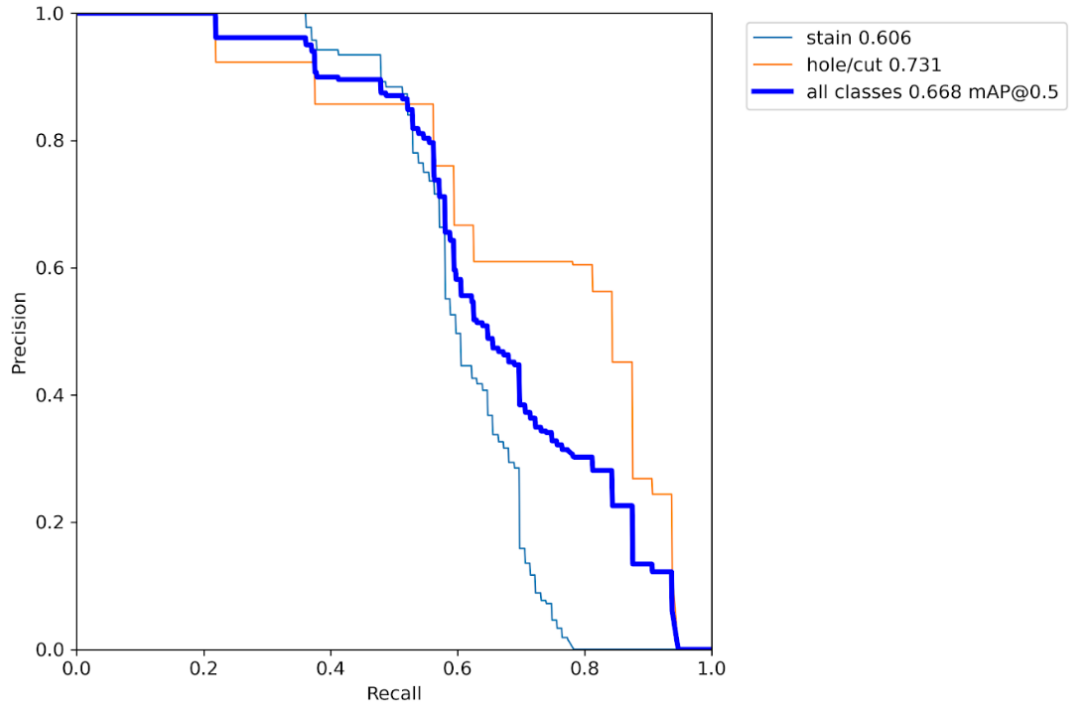
Precision (Kesinlik): Kesinlik, modelin doğru pozitiflerin oranını ifade eder. Yani, modelin pozitif olarak sınıflandırdığı örneklerin gerçekten ne kadarının pozitif olduğunu gösterir. Kesinlik, yanlış pozitiflerin (false positives) oluştuğu durumlarda düşük olabilir ve yanlış pozitiflerin oranını ifade eder. Yüksek kesinlik değeri, modelin pozitif olarak sınıflandırdığı örneklerin büyük bir kısmının gerçekten pozitif olduğunu gösterir.

Özetle, Recall (hatırlama) modelin ne kadar eksiksiz olduğunu, Precision (kesinlik) ise modelin ne kadar doğru olduğunu gösteren sınıflandırma performans ölçümleridir.



Şekil 3.17 : Recall - Confidence Grafiği

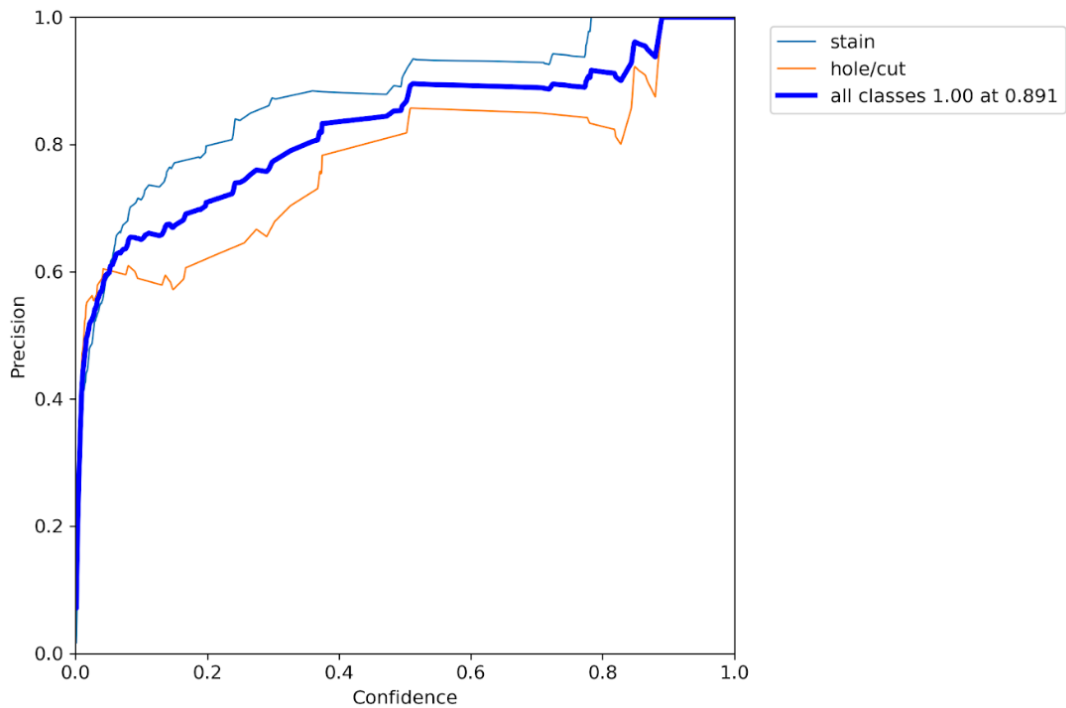
Recall - Confidence grafiği, sınıflandırma modelinin farklı güven eşiklerindeki performansını görsel olarak analiz etmemizi sağlar ve modelin kesinlik ve hatırlama arasındaki dengeyi nasıl sağladığını anlamamıza yardımcı olur.



Şekil 3.18 : Precision - Recall Grafiği

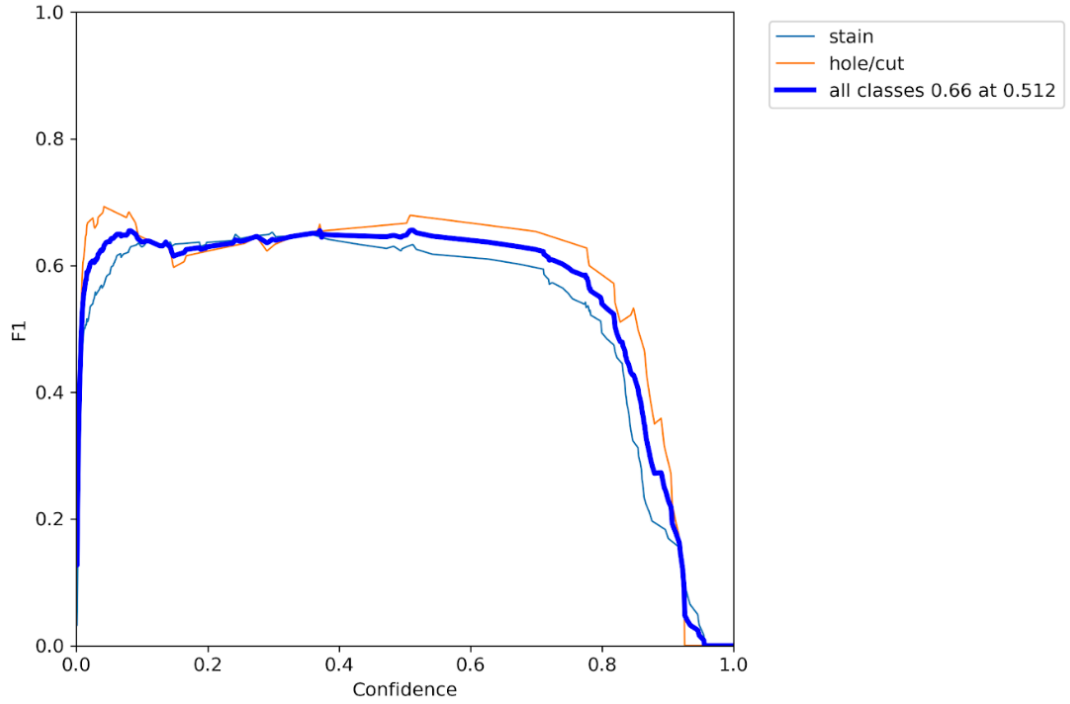
Bu grafik, sınıflandırma modelinin ne kadar doğru ve eksiksiz çalıştığını anlamak için kullanılır. Genellikle, kesinlik ve hatırlama arasındaki dengeyi sağlamak için grafikteki optimum noktayı seçmek istenir.

Precision - Recall grafiği, modelin performansını değerlendirmek ve sınıflandırma modelinin farklı eşik değerlerindeki performansını görselleştirmek için kullanılan etkili bir araçtır.



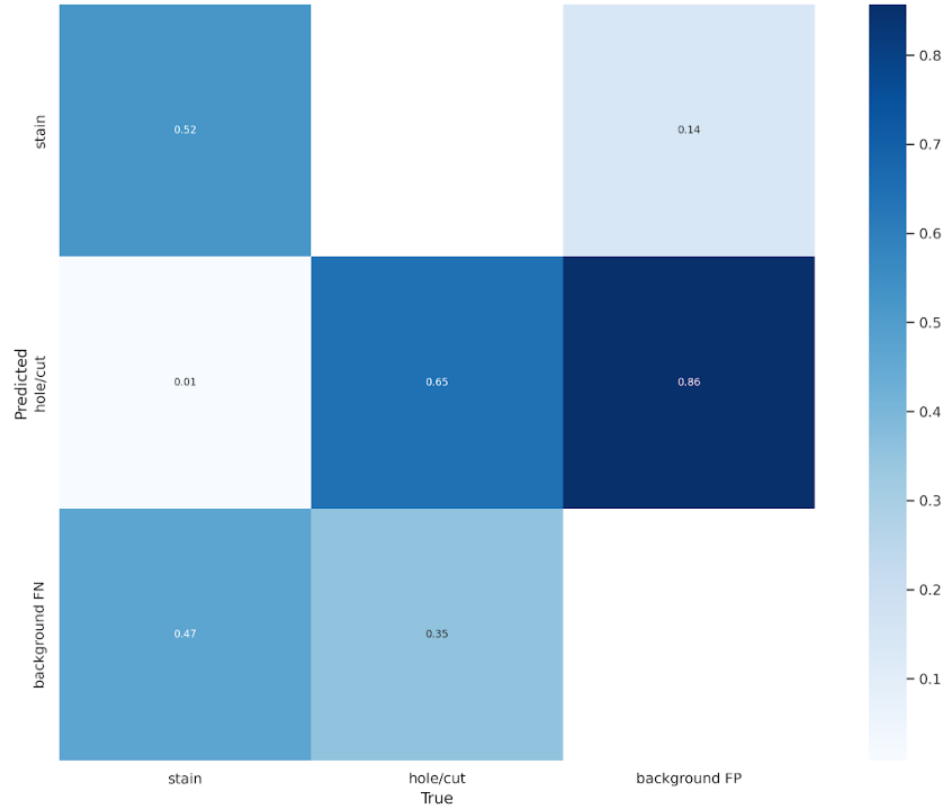
Şekil 3.19 : Precision - Confidence Grafiği

Precision - Confidence grafiği, bir sınıflandırma modelinin kesinlik (precision) değerlerini farklı güven (confidence) değerleri üzerinde gösteren bir grafikdir. Bu grafik, modelin güven değeri arttıkça kesinlik performansının nasıl değiştiğini görselleştirir. Grafikte, x-ekseni genellikle güven değerlerini, y-ekseni ise kesinlik değerlerini temsil eder. Grafikte yüksek güven değerlerine karşılık gelen kesinlik değerleri arasındaki ilişki incelenerek, modelin güven seviyesine göre ne kadar doğru çalıştığı değerlendirilebilir.



Şekil 3.20 : F1 - Confidence Grafiği

F1 - Confidence grafiği, bir sınıflandırma modelinin F1 skoru (F1 score) değerlerini farklı güven (confidence) değerleri üzerinde gösteren bir grafikdir. F1 skoru, kesinlik (precision) ve hatırlama (recall) değerlerinin harmonik ortalamasını ifade eder. Grafikte, x-ekseni genellikle güven değerlerini, y-ekseni ise F1 skoru değerlerini temsil eder. Grafikte yüksek güven değerlerine karşılık gelen yüksek F1 skorları arasındaki ilişki incelenerek, modelin güven seviyesine göre ne kadar dengeli bir performans gösterdiği değerlendirilebilir. F1 - Confidence grafiği, modelin kesinlik ve hatırlama arasındaki dengeyi göstererek, performans analizi yapmak ve en uygun güven seviyesini belirlemek için kullanılır.



Şekil 3.21 : Karmaşıklık Matrisi

Karmaşıklık matrisi (confusion matrix), bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir tablodur. Matris, gerçek sınıf etiketlerine göre modelin doğru ve yanlış sınıflandırdığı örneklerin sayılarını gösterir. Genellikle dört bölümden oluşur: doğru pozitifler (true positives), yanlış negatifler (false negatives), yanlış pozitifler (false positives) ve doğru negatifler (true negatives). Bu matris, modelin hangi sınıfı ne kadar doğru tahmin ettiğini, yanlış sınıflandırmaların nerede olduğunu ve performansını daha ayrıntılı bir şekilde analiz etmeyi sağlar. Karmaşıklık matrisi, modelin kesinlik (precision), hatırlama (recall), doğruluk (accuracy) ve F1 skoru (F1 score) gibi performans metriklerinin hesaplanmasında temel bir bileşendir.

3.9 API Oluşturulması

The image shows a code editor window with a dark background and light blue text. The code is for a Flask application. It starts with a route decorator for '/api/path' with the method 'POST'. Inside the route, there is a function 'create_user()' which takes 'request.get_data()' as input, decodes it from UTF-8, and then calls a function 'funcc(data)' to get a 'path'. Finally, it returns the 'path'. At the bottom, there is a standard Flask main block: 'if __name__ == '__main__': app.run(debug=True)'. The file name 'api.py' is visible in the top right corner of the editor window.

```
api.py

@app.route('/api/path', methods=['POST'])
def create_user():
    data = request.get_data() # İstekten veriyi al

    data= data.decode('UTF-8')
    path = funcc(data)

    return path

if __name__ == '__main__':
    app.run(debug=True)
```

Şekil 3.22 : Python'ı localhost'ta çalıştıran flask fonksiyonu

Bu kod, Flask web framework'ü kullanarak bir API endpoint oluşturur. Kodun başında, "/api/path" URL'si için POST isteği alacak olan " process_image" adında bir işlem tanımlanır. İstekten veriyi almak için request.get_data() yöntemi kullanılır ve alınan veri "data" değişkenine atanır. Daha sonra, "data" değişkeni "UTF-8" kodlamasına dönüştürülür. Sonrasında, "data" değişkeni "funcc" adlı bir işlevin parametresi olarak kullanılır ve bu işlevden dönen sonuç "path" değişkenine atanır.

En son olarak, "path" değişkeni yanıt olarak döndürülür.

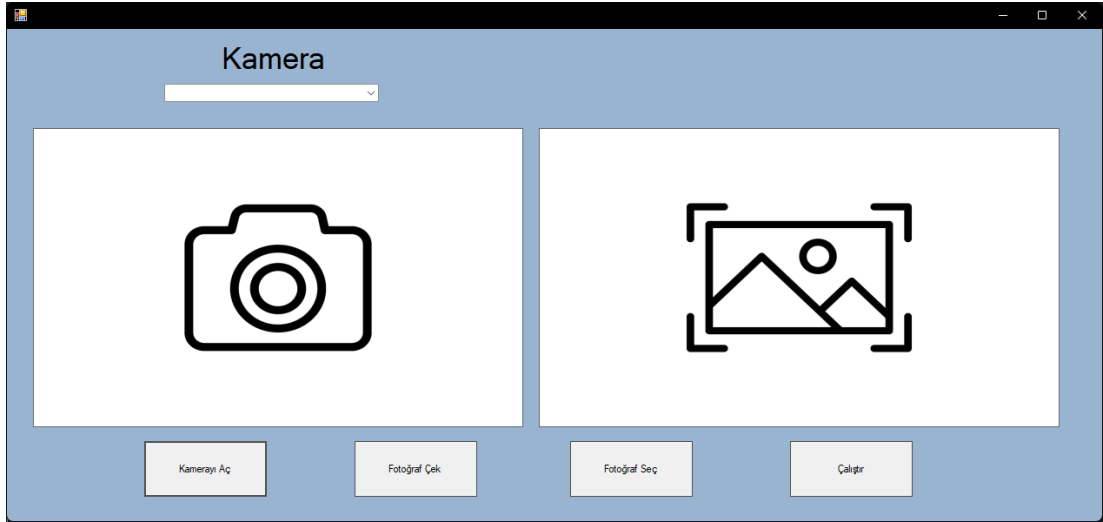
Kodun en altındaki if __name__ == '__main__': bloğu, bu dosyanın doğrudan çalıştırıldığında (diğer bir dosya tarafından ithal edilmediğinde) Flask uygulamasını çalıştırır. app.run(debug=True) komutu, Flask uygulamasını hata ayıklama modunda çalıştırır.

Bu kod, "/api/path" adresine gelen POST isteğini işleyen bir Flask API endpoint'i oluşturur. İstekten alınan veriyi işler, belirli bir işlevi çağırarak sonuç elde eder ve bu sonucu yanıt olarak döndürür.

4. UYGULAMA ÇIKTILARI

Bu bölümde atıflar, alıntılar ve dipnotların nasıl olması gerektiği hakkında bilgi verilecektir.

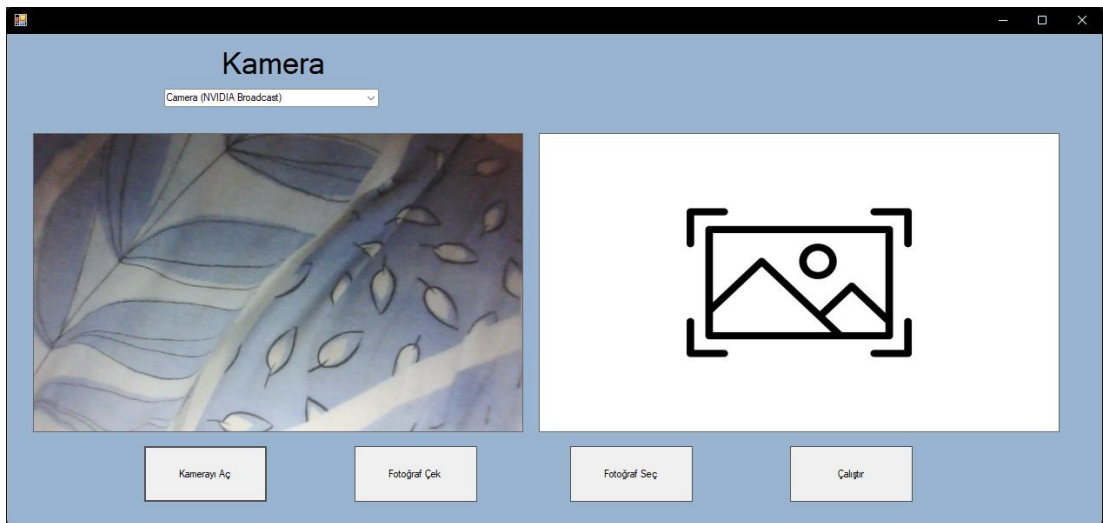
4.1 Uygulamanın Arayüzü



Şekil 4.1 : Uygulama Arayüzü

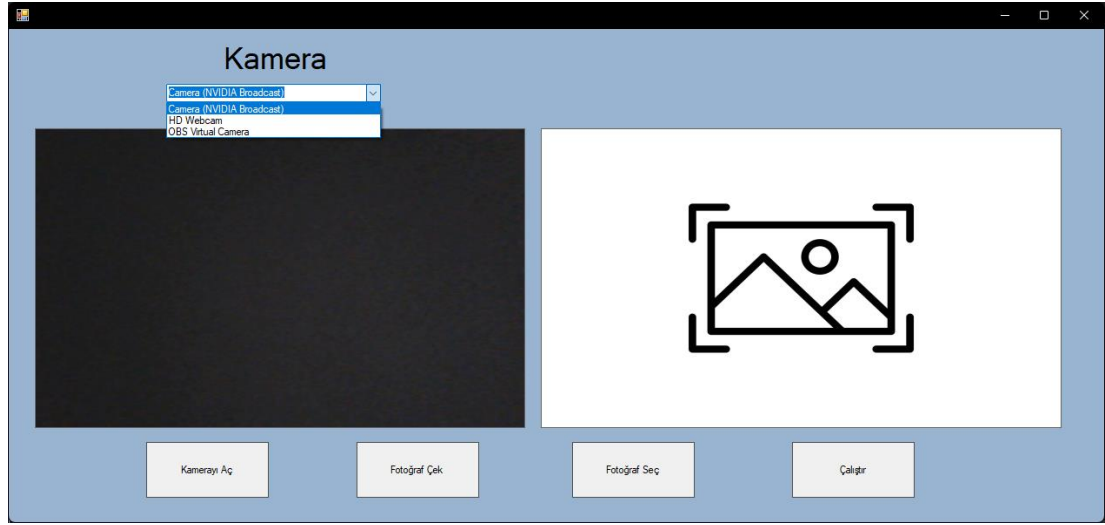
4.2 Kamera

Kamerayı Aç butonuna tıklandığında cihazınıza bağlı varsayılan kameradan görüntü almaya başlanacaktır.



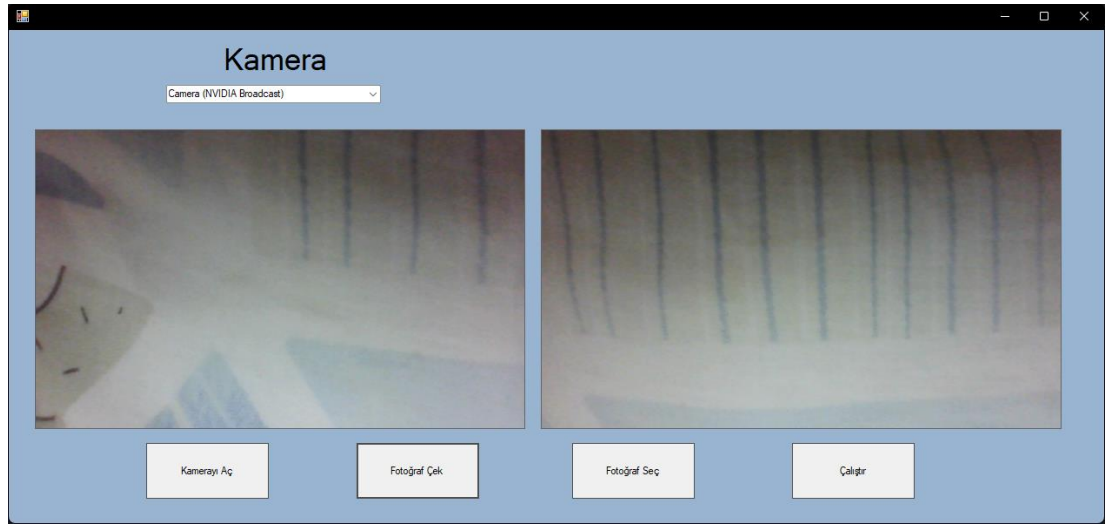
Şekil 4.2 : Kamera gösterilen bir kumaş görüntüsü

Eğer varsayılan kamerayı kullanmak istemiyorsanız Kamera başlığı altındaki seçenek menüsünden istediğiniz cihazı seçebilirsiniz.



Şekil 4.3 : Kamera seçme menüsü

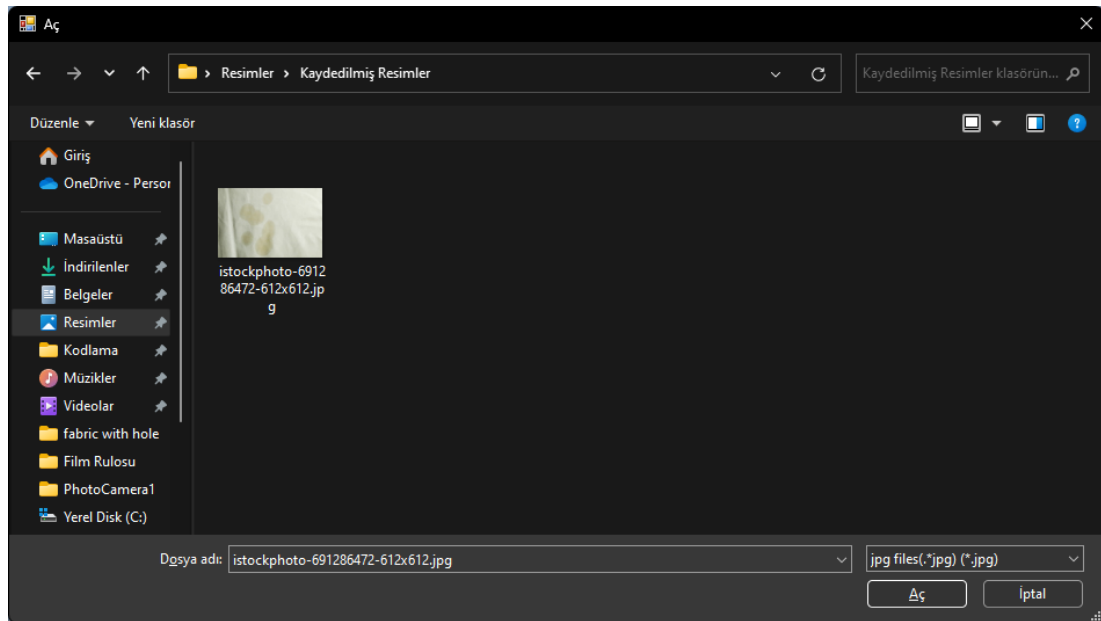
Kamerada istediğiniz görüntüyü aldıktan sonra fotoğraf çek butonuna basarak görseli kaydedebilir ve işleme hazır hale getirebilirsiniz.



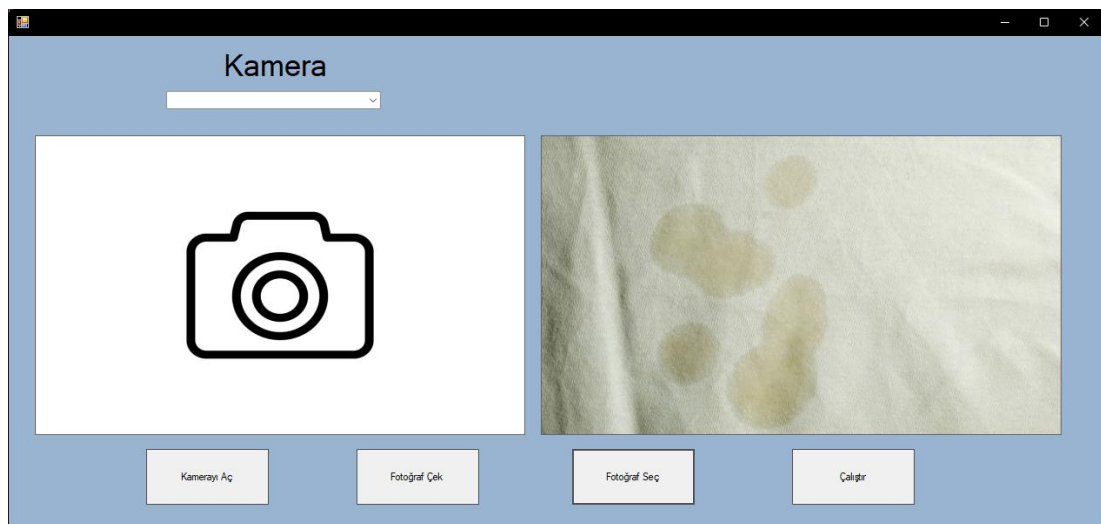
Şekil 4.4 : Soldaki görsel Fotoğraf çek butonu ile kaydedilip sağdaki kutuda görüntüleniyor

4.3 Fotoğraf Yükleme

Fotoraf Seç Butonuna basarak cihazınızda kaydedilmiş bir görseli seçebilirsiniz.



Şekil 4.5 : Gözet penceresi



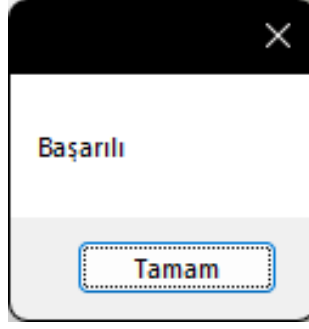
Şekil 4.6 : Seçilen Görsel

4.4 Çalıştır

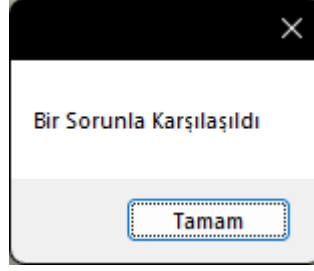
Çalıştır butonuna tıklandığında butona ait fonksiyonumuz çektiğimiz ya da hafızadan yüklediğimiz fotoğrafın konumunu Python API'sine POST isteği olarak gönderiyor.

Görüntü işleme Modelimiz görsel üzerinde işlemlerini yaptıktan sonra işlenmiş görselin konumunu geri döndürüyor.

Butona bastıktan sonra bir süre bekleyerek modelimizin bir değer döndürmesini bekliyoruz.

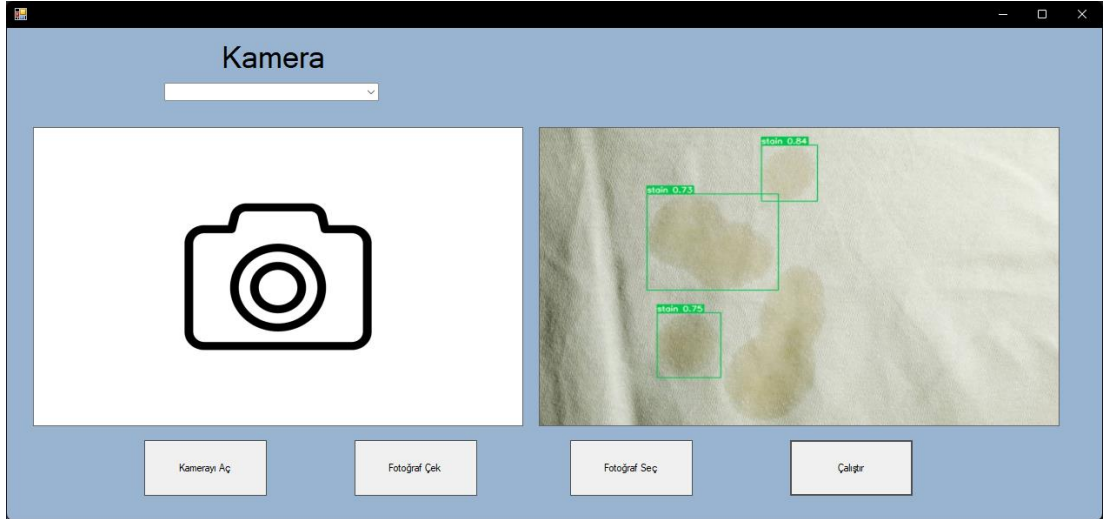


Şekil 4.7 : İşlem Başarılı ise Gelen Mesaj Kutusu



Şekil 4.8 : İşlem Başarısız ise gelen Mesaj Kutusu

Bu iki mesaj kutusundan birini gördüğümüzde işlem bitmiş demektir. Eğer başarılı mesajını aldıysak sağ taraftaki görselimiz python tarafından döndürülen görselle değiştirelecektir .



Şekil 4.9 : APT'den başarı ile dönen görsel

5. SONUÇ.

Bu proje, makine öğrenmesi ve bilgisayar görüşü tekniklerinin entegrasyonunu kullanarak kumaş hatalarının otomatik tespiti için bir yöntem geliştirmeyi amaçlamaktadır. Yapılan çalışmalar ve literatür taramaları, otomatik kumaş hata tespiti sistemlerinin, geleneksel yöntemlere kıyasla daha doğru, hızlı ve etkili sonuçlar sağladığını göstermektedir.

Projede kullanılan PyTorch gibi güçlü bir makine öğrenimi kütüphanesi, derin öğrenme modellerinin oluşturulması ve eğitilmesi için ideal bir araç sağlamaktadır. Bu sayede, yüksek çözünürlüklü kumaş görüntülerinden özelliklerin çıkarılması ve hatalı bölgelerin tespiti için derin öğrenme algoritmaları etkin bir şekilde kullanılabilir.

Proje sonucunda geliştirilen otomatik hata tespit sistemi, kumaş üretim endüstrisindeki kalite kontrol süreçlerini iyileştirmekte ve hataların erken tespit edilmesini sağlamaktadır. Gerçek zamanlı olarak kullanılabilen bu sistem, hatalı ürünlerin pazara sürülmesini engellerken, geri çağırma maliyetlerini azaltmakta ve üretkenliği artırmaktadır. Ayrıca, müşteri memnuniyetini artırarak marka itibarını koruma ve rekabet avantajı elde etme konusunda da önemli katkılar sağlamaktadır.

Bu projenin sonuçları, kumaş endüstrisi için değerli bir adım olarak kabul edilmelidir. Otomatik kumaş hata tespiti sisteminin kullanımı, üreticilere ve tüketicilere daha yüksek kaliteli ürünler sunma imkanı verirken, endüstrideki kalite kontrol standartlarının yükseltilmesine de katkı sağlamaktadır. Ayrıca, makine öğrenmesi ve bilgisayar görüşü tekniklerinin diğer endüstrilerde de kullanılması konusunda ilham verici olabilir.

Bundan sonra yapılacak çalışmalar, sistemin daha geniş veri setleriyle test edilmesi ve daha karmaşık kumaş türleri üzerindeki performansının değerlendirilmesini içermelidir. Ayrıca, sistemdeki hata tespitini daha da iyileştirmek için farklı özelliklerin ve algoritmaların kullanılması üzerine çalışmalar yapılabilir. Bu şekilde,

kumaş endüstrisindeki kalite kontrol süreçlerinin daha da optimize edilmesi ve hataların minimum düzeye indirilmesi hedeflenmektedir.

5.1 Çalışmanın Uygulama Alanı

Bu çalışma, kumaş endüstrisinde otomatik hata tespiti'nin önemli bir uygulama alanını ele almaktadır. Otomatik kumaş hata tespiti sistemleri, tekstil üreticileri ve tedarikçileri için büyük bir avantaj sağlamaktadır. Bu sistemler, kumaş üretim süreçlerinde kalite kontrolü iyileştirerek hataların erken tespit edilmesini ve düzeltilmesini sağlamaktadır.

Kumaş hataları, üretim sürecinde veya son ürünlerde oluşabilecek bir dizi sorunu içerebilir. Örnek olarak, yırtıklar, lekeler, renk uyumsuzlukları, desen hataları gibi durumlar verilebilir. Bu hatalar, üretici firmaların itibarını zedeler ve müşteri memnuniyetsizliğine yol açabilir. Bu nedenle, hataların hızlı bir şekilde tespit edilip düzeltilmesi, kumaş endüstrisi için kritik bir gereklilik haline gelmektedir.

Otomatik kumaş hata tespiti sistemleri, geleneksel yöntemlere kıyasla daha hızlı ve daha hassas sonuçlar sağlamaktadır. Bilgisayar görüşü ve derin öğrenme tekniklerinin kullanılmasıyla, yüksek çözünürlüklü kumaş görüntülerinden hatalı bölgelerin otomatik olarak tespit edilmesi mümkün hale gelmektedir. Bu sistemler, insan hatasını minimize ederken verimliliği artırır ve maliyetleri düşürür.

Bu çalışma, otomatik kumaş hata tespiti sistemlerinin kumaş üreticileri, tedarikçileri ve hatta son kullanıcılar için geniş bir uygulama potansiyeline sahip olduğunu göstermektedir. Bu sistemler, giyim endüstrisinde, ev tekstili üretiminde, otomotiv sektöründe ve daha birçok alanda kullanılabilir. Üreticiler, kalite kontrol süreçlerini iyileştirerek hataların minimum düzeye indirilmesini sağlayabilir ve sonuç olarak müşteri memnuniyetini artırabilir. Aynı zamanda, otomatik kumaş hata tespiti sistemleri, zaman ve maliyet tasarrufu sağlayarak üretkenliği artırır ve rekabet avantajı sağlar.

Bu çalışma, otomatik kumaş hata tespiti sistemlerinin kullanımının yaygınlaşmasıyla birlikte kumaş endüstrisindeki kalite kontrol standartlarının yükseltilmesine ve hatalı ürünlerin pazara sürülmesini önlemeye katkı sağlayacaktır. Ayrıca, makine öğrenmesi ve bilgisayar görüşü tekniklerinin diğer sektörlerdeki hata tespiti ve kalite kontrol uygulamalarında da yaygınlaşmasına örnek teşkil edecektir.

6. KAYNAKLAR

Flask Documentation. Erişim: 27 Mayıs 2023, <https://flask.palletsprojects.com/en/2.3.x/>

Matplotlib Documentation. Erişim: 17 Mayıs 2023, <https://matplotlib.org/stable/index.html>

OpenCV Documentation. Erişim: 15 Mayıs 2023, <https://docs.opencv.org/4.x/>

PyTorch Documentation. Erişim: 15 Mayıs 2023, <https://pytorch.org/docs/stable/index.html>

YoloV7 Erişim: 13 Mayıs 2023, <https://github.com/WongKinYiu/yolov7>

Ozkan, I., Kaya, H., & Polat, K. (2018). Deep learning framework for automatic fabric defect detection. *Measurement*, 125, 73-81.

Wu, X., Lu, Y., & Song, H. (2019). Fabric defect detection with convolutional neural networks. *IOP Conference Series: Materials Science and Engineering*, 541(1), 012012.

S. S. Kharade and S. R. Khot, "Analysis of Fabric Quality Using Deep Learning Approach," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Pune, India, 2020, pp. 1-5, doi: 10.1109/ICETITE51134.2020.9106289.

E. Kılınç and S. Ç. Çam, "Use of Convolutional Neural Network for Automatic Fabric Defect Detection," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2017, pp. 1-4, doi: 10.1109/IDAP.2017.8090357.

N. Aydın and H. Şahin, "Artificial Neural Networks for Fabric Defect Detection," 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2018, pp. 1-4, doi: 10.1109/SIU.2018.8404603.

K. Swathi and P. M. Deshpande, "Prediction of Fabric Quality using Artificial Neural Network," 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Vellore, India, 2019 [7]
Daniel Bacioiu, Geoff Melton,
Mayorkinos Papaelias, Rob Shaw (2019) Automated defect classification of SS304 TIG welding process using visible spectrum camera and machine learning, *NDT and E International* 107 (2019) 102139

Liu, Y., Liu, Z., & Wang, Y. (2018). Research on intelligent identification technology of textile fabric defect based on deep learning. *Journal of Physics: Conference Series*,

1094(1), 012097. doi: 10.1088/1742-6596/1094/1/012097