EÜ MÜHENDİSLİK FAKÜLTESİ

EGE ÜNİVERSİTESİ

LİSANS TEZİ

GÖRÜNTÜ İŞLEME İLE MOTİF ALGILAMA VE MOTİFİN YAPIM AŞAMALARINI KULLANICIYA SUNMA

Orkun EKE

Tez Danışmanı : Dr. Öğr. Üyesi Emine SEZER

Bilgisayar Mühendisliği Bölümü

Sunuş Tarihi : Haziran 2024

EGE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

(LİSANS TEZİ)

PATTERN RECOGNITION WITH IMAGE PROCESSING AND PRESENTING THE STAGES OF MAKING THE PATTERN TO THE USER

Orkun EKE

Tez Danışmanı: Dr. Öğr. Üyesi Emine SEZER

Bilgisayar Mühendisliği Bölümü

Bornova-İZMİR

2024

ÖZET

Görüntü İşleme ile Motif Algılama ve Motifin Yapım Adımlarını Kullanıcıya Sunma

EKE, Orkun

Lisans Tezi, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Dr. Öğr. Üyesi Emine SEZER

Haziran 2024, 44 Sayfa

Bu tez çalışmasında, tığ işi motiflerinin sınıflandırılması ve her bir motifin nasıl tığlanması gerektiğini otomatik olarak belirleyen bir desktop uygulama geliştirilmiştir. Uygulama, makine öğrenmesi ve ResNet teknolojisini ve algoritmalarını kullanarak motiflerin doğru bir şekilde tanınmasını ve sınıflandırılmasını sağlar. Çalışmada kullanılan veri seti, 10 farklı motif tipinde yaklaşık 5000 tığ işi resminden oluşmaktadır. Bu motif tipleri "Alpine tığ", "Feather tığ ", "Harlequin tığ ", "Herringbone tığ ", "Jasmine tığ ", "Moss tığ ", "Shell tığ ", "Star tığ ", "Waffle tığ " olarak belirlenmiştir. Tezde öncelikle, kullanılan veri seti ve bu veri setinin genişletilmesi için uygulanan veri artırma yöntemleri açıklanmıştır. Daha sonra, farklı makine öğrenmesi algoritmaları denenmiş ve en iyi performans gösteren model seçilmiştir. Uygulamada Python programlama dili ve çeşitli kütüphaneler kullanılarak bir kullanıcı arayüzü oluşturulmuştur. Bu arayüz, kullanıcıların motif resimlerini yükleyip sınıflandırma sonuçlarını ve tığlama talimatlarını alabilecekleri şekilde tasarlanmıştır. Bu çalışma, tığ işi motiflerinin otomatik tanınması ve doğru tığlama talimatlarının sunulması konusunda önemli bir katkı sağlamaktadır. Gelecekte, bu tür uygulamaların daha fazla motif tipi eklenerek ve kullanıcı deneyimi iyileştirilerek genişletilmesi hedeflenmektedir.

Anahtar Sözcükler: Tığ işi, Makine Öğrenmesi, Görüntü Sınıflandırma, Python, Veri Artırma, ResNet

ABSTRACT

Implementation and Analysis of a Decision-Making Algorithm Using Game Theory, Fuzzy Logic, and Dynamic Programming in Air Combat

EKE, Orkun

Undergraduate Thesis, Department of Computer Engineering

Thesis Advisor: Assistant Professor Dr. Emine SEZER

June 2024, 44 Pages

This thesis presents the development of a desktop application that automatically classifies crochet motifs and provides instructions on how to crochet each identified motif. The application utilizes machine learning algorithms and ResNet techonlogy to accurately recognize and categorize the motifs. The dataset used in this study comprises approximately 5000 crochet images categorized into 10 different motif types: "Alpine crochet", "Feather stitch", "Harlequin Stitch", "Herringbone crochet", "Jasmine crochet", "Moss crochet", "Shell crochet", "Star stitch", and "Waffle stitch". Initially, the thesis describes the dataset and the data augmentation techniques applied to expand it. Subsequently, different machine learning algorithms were tested, and the model with the best performance was selected. A user interface was created using the Python programming language and various libraries, allowing users to upload motif images and receive classification results and crocheting instructions. At the conclusion of the thesis, the performance of the developed model was evaluated using metrics such as classification accuracy and misclassification rate. This work makes a significant contribution to the automatic recognition of crochet motifs and the provision of accurate crocheting instructions. Future improvements aim to include more motif types and enhance the user experience of such applications.

Keywords: Crochet, Machine Learning, Image Classification, Python, Data Augmentation, ResNet

TEŞEKKÜRLER

Lisans tezimizi hazırlarken; bana yol gösteren, desteğini ve yardımlarını esirgemeyen akademik danışmanımız Sayın Dr. Öğr. Üyesi Emine SEZER'e ve eğitim hayatım boyunca hem maddi hem de manevi olarak bizi destekleyen aileme ve arkadaşlarıma teşekkürü borç bilirim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KISALTMALAR DİZİNİ

API	Application Programming Interface
BMP	Bitmap
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GIF	Graphics Interchange Format
GUI	Graphical User Interface
IDE	Integrated Development Environment
JPG/JPEG	Joint Photographic Experts Group
LSTM	Long Short Term Memory
NLP	Natural Language Processing
PIL	Python Imaging Library
PNG	Portable Network Graphics
RNN	Recurrent Neural Network
TIFF	Tagged Image File Format
TPU	Tensor Processing Unit
YZ	Yapay Zeka

1. GİRİŞ

Tığ işi, dünya genelinde yaygın olarak yapılan ve köklü bir geçmişe sahip olan bir el sanatıdır. Tığ işinin tam olarak nerede ve ne zaman ortaya çıktığı kesin olarak bilinmemekle birlikte, bazı kaynaklar bu tekniğin Güney Amerika, Arabistan ve Çin gibi bölgelerde bağımsız olarak gelişmiş olabileceğini öne sürmektedir [1]. 1800'lü yıllara gelindiğinde, günümüzde bildiğimiz tığ işi tekniği Avrupa'da yaygınlaşmıştır. Fransızca'da "kanca" anlamına gelen "crochet" terimi, bu tekniğin adı olarak benimsenmiştir [2].

Tığ işi, özellikle 19. yüzyılda Avrupa'da büyük bir popülerlik kazanmıştır. Viktorya döneminde, tığ işi zarif kıyafetler ve ev eşyaları yapmak için kullanılıyordu. Bu dönemde yayımlanan desen kitapları, tığ işinin yaygınlaşmasına ve ev hanımları arasında yaygın bir hobi haline gelmesine katkı sağladı [3]. Özellikle İrlanda'da ortaya çıkan İrlanda danteli, bu dönemin en bilinen tığ işi örneklerinden biridir. İrlanda'daki patates kıtlığı sırasında, tığ işi dantel üretimi ailelerin geçimini sağlamak için önemli bir gelir kaynağı olmuştur [4].

20. yüzyılın başlarında, tığ işi daha çok günlük kıyafetler ve battaniyeler yapmak için kullanılmıştır. Birinci Dünya Savaşı sırasında, askerler için çorap, şapka ve şal gibi ürünlerin örülmesi, tığ işinin vatanseverlik simgesi haline gelmesine neden olmuştur. 1940'lar ve 50'ler boyunca, tığ işi daha kitsch bir trende sahip olup kutulu takımlar ve süslü elbiseler için chenille iplik kullanılmıştır. 1960'larda ise tığ işi, hippi ve anti-kuruluş estetiği ile özdeşleşmiş ve özgür formdaki, psikedelik motiflerle dolu "tığ işi devrimi" yaşanmıştır [1].

21. yüzyıla gelindiğinde, tığ işi yeniden popülerlik kazanmış ve modern tasarımlar ile daha canlı renkler kullanılmaya başlanmıştır. İnternetin gelişmesi ile birlikte, tığ işi hakkında bilgi edinmek, desenlere ulaşmak ve öğrenmek çok daha kolay hale gelmiştir. Ravelry ve Etsy gibi web siteleri, tığ işi yapanların çalışmalarını tanıtmaları ve satmaları için platformlar sunmaktadır [5]. Bugün, tığ işi hem kişisel ifade biçimi olarak hem de meditasyon ve stres azaltma aracı olarak değer görmektedir [2].

Görüntü işleme, dijital görüntüler üzerinde çeşitli işlemler yaparak bu görüntülerin analizi ve iyileştirilmesi sürecidir. Bu alan, tıbbi görüntüleme, yüz tanıma, otonom araçlar ve güvenlik sistemleri gibi birçok farklı uygulama alanında kullanılmaktadır [6][7]. Görüntü işleme teknikleri, nesne tespiti, segmentasyon, sınıflandırma ve özellik çıkarımı gibi adımları içerebilir. Makine öğrenmesi ve özellikle derin öğrenme algoritmaları, görüntü işleme alanında büyük ilerlemeler kaydetmiş ve bu tekniklerin daha doğru ve hızlı uygulanmasına olanak tanımıştır [8][9].

Görüntü işleme, genellikle birkaç temel adımdan oluşur; Ön İşleme: Görüntünün iyileştirilmesi ve gürültünün azaltılması. Segmentasyon: Görüntünün farklı bileşenlere ayrılması. Özellik Çıkarımı: Görüntüdeki önemli özelliklerin belirlenmesi. Sınıflandırma: Görüntünün belirli kategorilere ayrılması [8][9]. Bu teknikler, özellikle Convolutional Neural Networks (CNNs) gibi derin öğrenme algoritmaları ile birlikte kullanıldığında yüksek doğrulukla sonuçlar elde edilebilir. CNN'ler, görüntülerdeki desenleri tanıma ve sınıflandırma konusunda oldukça etkilidir [10][11].

Bu projenin amacı, tığ işi motiflerini doğru bir şekilde tanıyabilen ve sınıflandırabilen bir desktop uygulama geliştirmektir. Kullanıcılar, bu uygulama aracılığıyla motif resimlerini yükleyip sınıflandırma sonuçlarını ve tığlama talimatlarını alabilirler. Projede kullanılan veri seti, 10 farklı motif tipinde yaklaşık 5000 tığ işi resminden oluşmaktadır. Bu motif tipleri "alpine crochet", "feather stitch", "harlequin Stitch", "herringbone crochet", "jasmine crochet", "moss crochet", "shell crochet", "star stitch", "waffle stitch" olarak belirlenmiştir.

2. LİTERATÜR ÇALIŞMASI

Bu çalışmada, tığ işi motiflerinin otomatik olarak tanınması ve sınıflandırılması amacıyla bir derin öğrenme modeli eğitilmiştir. Modelin eğitim süreci, veri hazırlama, model seçimi, eğitim ve değerlendirme aşamalarını içermektedir.

İlk adım, geniş ve çeşitli bir veri seti hazırlamaktır.. Projede niş bir veri türü seçildiği için veri taramasında hazır bir veri sistemi bulunmamıştır. Önce genel bir tığ motif araştırılması yapılmıştır. Eğitim için öncelikli olan miktar-kalite orantısına göre 10 farklı tip seçilip 50 farklı kaynaktan 466 ayrı veri toplanmıştır. Bu veriler model eğitimi için çeşitli algoritmalardan geçip (Data augmentation) 5000 civarı veri elde edilmiştir. Veri artırma teknikleri kullanılarak veri seti genişletilmiş ve çeşitlendirilmiştir. Bu teknikler arasında döndürme, ölçeklendirme, kırpma ve renk değiştirme gibi yöntemler bulunmaktadır .

Tığ işi motiflerini tanımak ve sınıflandırmak için Convolutional Neural Networks (CNNs) kullanılmıştır. CNN'ler, görüntü işleme ve sınıflandırma görevlerinde yüksek doğruluk ve verimlilik sağlar. Model, TensorFlow ve Keras gibi popüler derin öğrenme kütüphaneleri kullanılarak geliştirilmiştir. Bu kütüphaneler, modelin oluşturulması, eğitilmesi ve değerlendirilmesi için güçlü araçlar sunar .

Eğitim sürecinde, veri seti eğitim ve test olmak üzere ikiye ayrılmıştır. Model, eğitim verileri üzerinde eğitilmiş ve test verileri ile doğrulanmıştır. Eğitim sırasında, modelin öğrenme oranı, epoch sayısı ve batch boyutu gibi hiperparametreler optimize edilmiştir. Ayrıca, modelin aşırı uyumunu önlemek için dropout ve erken durdurma gibi düzenleme teknikleri kullanılmıştır .

Bu çalışmada kullanılan başlıca teknolojiler ve araçlar şunlardır. Python: Modelin geliştirilmesi ve eğitimi için kullanılan programlama dili. TensorFlow ve Keras: Derin öğrenme modellerinin oluşturulması ve eğitimi için kullanılan kütüphaneler. OpenCV: Görüntü işleme görevleri için kullanılan açık kaynaklı kütüphane. Google Colab: Model eğitimi ve deneyler için kullanılan bulut tabanlı platformdur .

3. TEKNOLOJİLER

Teknolojiler bölümünde projenin yapımında kullanılan teknolojiler sıralanakcak ve genel olarak anlatılacaktır. Projede kullanılan teknojilerin önemi ve yerine göre detaylı açıklamalar getilecektir.

3.1 Python

Python, Guido van Rossum tarafından 1991 yılında geliştirilen, yüksek seviyeli, genel amaçlı bir programlama dilidir. Python, okunabilirlik ve sadelik üzerine odaklanmış bir dil olup, kullanıcı dostu sözdizimi sayesinde kod yazımını kolaylaştırır. Bu özellikleri, Python'u hem yeni başlayanlar hem de deneyimli programcılar arasında popüler hale getirmiştir. Python'un diğer avantajları arasında geniş standart kütüphane desteği, açık kaynak olması ve çok çeşitli alanlarda kullanılabilirliği bulunmaktadır [12].

Python, bilimsel hesaplamalar, veri analizi, yapay zeka ve makine öğrenmesi gibi alanlarda sıkça tercih edilmektedir. Ayrıca, web geliştirme, otomasyon ve yazılım geliştirme gibi çeşitli uygulama alanlarında da yaygın olarak kullanılmaktadır. Python'un esnekliği ve geniş kütüphane desteği, karmaşık projelerin hızlı ve verimli bir şekilde geliştirilmesine olanak tanır.[12]

Bu projede yüksek dil olarak python’ın kullanılması IDE’nin getirdiği esneklik ve çeşitli kütüphanelerdir. Bu projede Pyton’a ait 4 adet farklı kütüphane aynı zamanda Python ile uyumlu 2 ayrı kütüphane daha kullanılmıştır.

3.2 Genel Anlamıyla Yapay Zeka

Yapay zeka, insan zekasını taklit eden ve insan benzeri görevleri yerine getirebilen sistemlerin geliştirilmesiyle ilgilenen bir bilgisayar bilimi alanıdır. Bu sistemler, öğrenme, problem çözme, muhakeme ve karar verme gibi bilişsel işlevleri yerine getirebilir. YZ, günümüzde birçok farklı alanda kullanılmaktadır ve gelecekte daha da önemli bir rol oynaması beklenmektedir. [13] YZ'nin Temel Kavramları:

- Öğrenme: YZ sistemleri, veri ve deneyimlerden öğrenerek zamanla performanslarını geliştirebilirler.
- Problem Çözme: YZ sistemleri, karmaşık problemleri analiz ederek ve en uygun çözümleri bularak çözebilirler.
- Muhakeme: YZ sistemleri, mantıksal ve olasılıksal çıkarımlar yaparak yeni bilgilere ulaşabilirler.
- Karar Verme: YZ sistemleri, topladıkları bilgileri kullanarak bilinçli kararlar verebilirler.

3.2.1 Makine Öğrenmesi

Makine öğrenmesi, bilgisayarların veri üzerinden öğrenmesini ve belirli görevleri otomatik olarak gerçekleştirmesini sağlayan bir yapay zeka alt alanıdır. Makine öğrenmesi, algoritmalar ve istatistiksel modeller kullanarak verilerdeki kalıpları tanır ve tahminler yapar. Bu süreç, veri toplama, veri temizleme, model eğitimi ve model değerlendirme adımlarını içerir [13].

- Denetimli Öğrenme: Etiketli veri setleri kullanılarak model eğitimi yapılır. Bu yöntem, sınıflandırma ve regresyon gibi görevler için kullanılır
- Denetimsiz Öğrenme: Etiketlenmemiş veri setleri kullanılarak model eğitimi yapılır. Bu yöntem, kümeleme ve boyut indirgeme gibi görevler için kullanılır.
- Pekiştirmeli Öğrenme: Ajanlar, çevreleriyle etkileşimde bulunarak ödül ve ceza mekanizmaları ile öğrenirler. Bu yöntem, oyunlar ve otonom sistemler gibi uygulamalarda kullanılır [14].

3.2.2 Derin Öğrenme

Derin öğrenme, makine öğrenmesinin bir alt dalıdır ve yapay sinir ağları kullanılarak öğrenme işlemini gerçekleştirir. Derin öğrenme modelleri, genellikle çok katmanlı sinir ağları kullanarak verilerdeki karmaşık kalıpları tanır ve tahminler yapar. Bu modeller, büyük veri setleri ve güçlü hesaplama kaynakları gerektirir [13].

Derin öğrenme, görüntü tanıma, ses tanıma, doğal dil işleme ve oyunlar gibi birçok alanda başarıyla uygulanmaktadır. Derin öğrenme algoritmaları, özellikle Convolutional

Neural Networks (CNNs) ve Recurrent Neural Networks (RNNs) gibi mimariler üzerine kuruludur:

- Convolutional Neural Networks (CNNs): Görüntü işleme ve bilgisayarla görme görevlerinde yaygın olarak kullanılır. CNN'ler, verilerdeki uzamsal hiyerarşileri tanımak için katmanlı yapılar kullanır.
- Recurrent Neural Networks (RNNs): Zaman serisi verileri ve doğal dil işleme görevlerinde kullanılır. RNN'ler, ardışık veri noktaları arasındaki bağımlılıkları öğrenmek için geri besleme döngüleri kullanır [15].

3.3 TensorFlow

TensorFlow, Google tarafından geliştirilen ve 2015 yılında açık kaynak olarak yayınlanan bir makine öğrenme kütüphanesidir. TensorFlow, derin öğrenme modellerinin geliştirilmesi ve eğitilmesi için geniş çapta kullanılan güçlü bir araç seti sunar. TensorFlow'un esnek mimarisi, araştırmacılar ve geliştiriciler için model oluşturma sürecini kolaylaştırır ve hızlandırır.

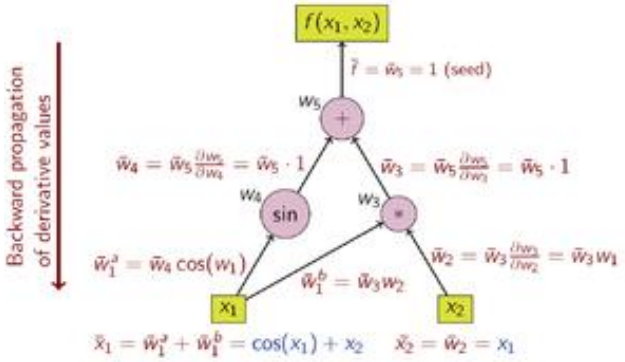
TensorFlow, hesaplama grafikleri kullanarak sayısal hesaplamalar yapar. Bu grafikler, düğümler (nodes) ve kenarlardan (edges) oluşur; düğümler matematiksel işlemleri temsil ederken, kenarlar bu işlemler arasındaki veri akışını ifade eder. Bu yapı, özellikle karmaşık sinir ağları ve derin öğrenme modelleri için uygun bir ortam sağlar [14].

TensorFlow'un temel yapı taşı, hesaplama grafikleridir. Hesaplama grafikleri, çeşitli matematiksel işlemleri ve veri akışını temsil eden yönlendirilmiş grafiklerdir. Bu grafikleri kullanarak, TensorFlow karmaşık hesaplamaları verimli bir şekilde gerçekleştirebilir. Hesaplama grafikleri, TensorFlow'un esnekliğini ve performansını artıran önemli bir özelliktir [14].

TensorFlow'un ismi, "tensor" kelimesinden gelir. Tensor, çok boyutlu bir diziyi ifade eder. TensorFlow, verileri tensorlar olarak işler ve bu tensorlar üzerinden hesaplamalar yapar. Bu yapı, özellikle büyük veri setleri ve yüksek boyutlu verilerle

çalışırken önemli avantajlar sağlar. Tensorlar, skaler (tek boyutlu), vektör (iki boyutlu) ve matris (üç boyutlu) gibi çeşitli şekillerde olabilir [15].

Otomatik Diferansiyon: Makine öğrenme modellerinin eğitimi sırasında, genellikle gradyan inişi gibi optimizasyon algoritmaları kullanılır. Bu algoritmalar, modelin performansını iyileştirmek için hata fonksiyonunun türevlerini hesaplar. TensorFlow, otomatik diferansiyasyon (automatic differentiation) adı verilen bir teknik kullanarak bu türevleri otomatik olarak hesaplar. Bu özellik, model eğitimi sürecini basitleştirir ve hızlandırır [14].



Şekil 3.1 Otomatik Diferansiyon Hesaplama Algoritması

TensorFlow, geniş bir uygulama yelpazesine sahiptir. Başlıca kullanım alanları şunlardır:

- Derin Öğrenme: Sinir ağları ve derin öğrenme modellerinin oluşturulması ve eğitilmesi için yaygın olarak kullanılır
- Doğal Dil İşleme (NLP): Metin sınıflandırma, duygu analizi, dil modeli oluşturma gibi NLP görevlerinde kullanılır.
- Bilgisayarla Görme: Görüntü sınıflandırma, nesne tespiti, yüz tanıma gibi bilgisayarla görme uygulamalarında kullanılır.
- Öneri Sistemleri: Kullanıcı tercihlerini analiz ederek kişiselleştirilmiş öneriler sunan sistemlerde kullanılır.
- Otonom Sistemler: Otonom araçlar ve robotlar için karar verme ve hareket planlama görevlerinde kullanılır [15].

TensorFlow’u avantajları ve dezavantajları olarak değerlendirecek olursak. Avantajları:

- Esneklik: TensorFlow, çeşitli platformlarda (CPU, GPU, TPU) çalışabilir ve genişletilebilir bir yapıya sahiptir.
- Geniş Topluluk Desteği: TensorFlow, geniş bir kullanıcı topluluğu ve kapsamlı dökümantasyon ile desteklenir. Bu, geliştiricilerin karşılaştıkları sorunları çözmelerini ve projelerini hızla ilerletmelerini sağlar.
- Performans: TensorFlow, büyük veri setleri ve karmaşık modellerle çalışırken yüksek performans sunar. GPU ve TPU desteği, hesaplama hızını önemli ölçüde artırır [15].

Dezavantajlarına bakılacak olursa:

- Geniş Topluluk Desteği: TensorFlow, geniş bir kullanıcı topluluğu ve kapsamlı dökümantasyon ile desteklenir. Bu, geliştiricilerin karşılaştıkları sorunları çözmelerini ve projelerini hızla ilerletmelerini sağlar.
- Karmaşıklık: TensorFlow, öğrenmesi ve kullanması zor olabilen karmaşık bir kütüphanedir. Yeni başlayanlar için eğitimi ve model geliştirmeyi öğrenmek zaman alabilir.
- Bellek Kullanımı: TensorFlow, büyük veri setleri ve karmaşık modellerle çalışırken yüksek bellek kullanımı gerektirebilir. Bu durum, bellek yönetimi ve optimizasyon gerektirebilir [16].

TensorFlow, bu adımları kolaylaştıran ve hızlandıran çeşitli araçlar ve fonksiyonlar sunar. TensorFlow’un esnek yapısı ve güçlü hesaplama yetenekleri, onu makine öğrenme ve derin öğrenme projelerinde vazgeçilmez bir araç haline getirir.

3.4 Keras

Keras, yüksek seviyeli bir yapay sinir ağı API'sidir ve makine öğrenmesi modellerini kolay ve hızlı bir şekilde oluşturmak için kullanılır. François Chollet tarafından geliştirilmiş olup, 2015 yılında açık kaynak olarak yayınlanmıştır. Keras, kullanıcı dostu arayüzü ile karmaşık derin öğrenme modellerini bile basitçe tanımlama ve eğitme olanağı sağlar. TensorFlow, Microsoft Cognitive Toolkit (CNTK) ve Theano gibi

çeşitli arka uçlar üzerinde çalışabilir. Bu esneklik, Keras'ın popülerliğini artırmış ve geniş bir kullanıcı kitlesi kazanmasına yardımcı olmuştur [17]. Keras'ın kullanılmasını sağlayan temel özellikleri sıralanmıştır.

- Modülerlik: Keras, tamamen modüler bir yapıya sahiptir. Sinir ağlarının katmanları, aktivasyon fonksiyonları, optimizasyon algoritmaları ve kayıp fonksiyonları gibi bileşenler birbirinden bağımsız olarak kullanılabilir ve birleştirilebilir.
- Kullanıcı Dostu: Keras, öğrenmesi ve kullanması kolay bir API sunar. Bu, yeni başlayanlar için derin öğrenme modellerini hızlı bir şekilde oluşturma ve eğitme sürecini kolaylaştırır.
- Esneklik: Keras, çeşitli arka uçlar (backend) üzerinde çalışabilir. TensorFlow, Theano ve Microsoft Cognitive Toolkit (CNTK) gibi arka uçlar arasında kolayca geçiş yapılabilir.
- Taşınabilirlik: Keras, modellerin kolayca taşınabilir ve dağıtılabilir olmasını sağlar. Bu, modellerin farklı platformlarda (örneğin mobil cihazlar ve web sunucuları) çalıştırılabilmesini mümkün kılar [17].

Keras'ın kullanım alanları aşağıda sıralanmıştır:

- Görüntü İşleme: Keras, Convolutional Neural Networks (CNNs) kullanarak görüntü sınıflandırma, nesne tespiti ve segmentasyon gibi görevlerde kullanılır.
- Doğal Dil İşleme (NLP): Metin sınıflandırma, duygu analizi, dil modeli oluşturma ve metin üretimi gibi NLP görevlerinde kullanılır.
- Zaman Serisi Analizi: Recurrent Neural Networks (RNNs) ve Long Short-Term Memory (LSTM) ağları kullanarak zaman serisi tahmini ve analizinde kullanılır.

Keras'ı kullanmanın avantajları aşağıda sıralanmıştır:

- Zaman Serisi Analizi: Recurrent Neural Networks (RNNs) ve Long Short-Term Memory (LSTM) ağları kullanarak zaman serisi tahmini ve analizinde kullanılır.
- Kolay Kullanım: Keras, kullanıcı dostu ve öğrenmesi kolay bir API sunar. Bu, hızlı prototipleme ve model geliştirme süreçlerini hızlandırır.

- Esneklik ve Modülerlik: Keras, çeşitli arka uçlar üzerinde çalışabilen esnek ve modüler bir yapıya sahiptir. Bu, farklı projeler ve uygulamalar için kolayca uyarlanabilir.
- Geniş Topluluk ve Destek: Keras, geniş bir kullanıcı topluluğuna ve kapsamlı dökümantasyon ile desteklenir. Bu, kullanıcıların karşılaştıkları sorunları çözmelerini ve projelerini hızla ilerletmelerini sağlar [17].

Keras'ın dezavantajları aşağıda sıralanmıştır:

- Performans Kısıtlamaları: Keras, yüksek performans gerektiren bazı ileri seviye derin öğrenme uygulamalarında sınırlamalar gösterebilir. Bu durumlarda, doğrudan TensorFlow veya diğer arka uç kütüphanelerinin kullanılması gerekebilir.
- Azaltılmış Esneklik: Keras, kullanıcı dostu ve basit bir API sağlamak amacıyla bazı ileri seviye yapılandırma seçeneklerini sınırlayabilir. Bu, karmaşık modeller veya özelleştirilmiş işlemler için esneklik kaybına yol açabilir [18].

3.5 OpenCV

OpenCV (Open Source Computer Vision Library), bilgisayarla görme ve görüntü işleme alanlarında kullanılan açık kaynaklı bir kütüphanedir. 1999 yılında Intel tarafından geliştirilmiş olup, daha sonra Willow Garage ve Itseez tarafından desteklenmiştir. OpenCV, gerçek zamanlı görüntü işleme görevlerini gerçekleştirmek için geniş bir araç seti sunar ve Python, C++, Java gibi birçok programlama dili ile uyumludur [19].

OpenCV, görüntü işleme ve bilgisayarla görme uygulamalarında yaygın olarak kullanılan birçok özellik ve fonksiyonu içerir. Bu özelliklerden bazıları şunlardır:

- Görüntü İşleme: OpenCV, görüntü okuma, yazma, dönüştürme, filtreleme ve geometrik dönüşümler gibi temel görüntü işleme fonksiyonlarını destekler.
- Nesne Tespiti: OpenCV, Haar kaskat sınıflandırıcıları, HOG (Histogram of Oriented Gradients) ve derin öğrenme tabanlı yöntemler kullanarak nesne tespiti gerçekleştirebilir.
- Özellik Çıkarımı ve Eşleme: OpenCV, SIFT, SURF, ORB gibi çeşitli algoritmalar kullanarak görüntü özelliklerini çıkarabilir ve bu özellikleri eşleştirebilir.

- Hareket Algılama: OpenCV, arka plan çıkarma ve optik akış gibi tekniklerle hareket algılama ve takibi yapabilir.
- Makine Öğrenmesi: OpenCV, k-ortalama, karar ağaçları, rastgele ormanlar ve destek vektör makineleri (SVM) gibi çeşitli makine öğrenmesi algoritmalarını içerir [19].

OpenCV, birçok farklı uygulama alanında kullanılmaktadır. Başlıca kullanım alanları şunlardır:

- Görüntü ve Video İşleme: OpenCV, görüntü ve video dosyalarının işlenmesi ve analizi için kullanılır. Bu, görüntü filtreleme, nesne tespiti, yüz tanıma ve hareket algılama gibi görevleri içerir.
- Bilgisayarla Görme: OpenCV, robotik ve otonom sistemlerde bilgisayarla görme görevlerini gerçekleştirmek için yaygın olarak kullanılır. Bu, nesne algılama, konumlandırma ve takip gibi görevleri içerir.
- Tıbbi Görüntüleme: OpenCV, tıbbi görüntüleme ve analiz uygulamalarında da kullanılır. Bu, MRI ve CT taramalarının işlenmesi, anormalliklerin tespiti ve segmentasyon gibi görevleri içerir.
- Endüstriyel Otomasyon: OpenCV, kalite kontrol, hata tespiti ve otomatik montaj hatları gibi endüstriyel otomasyon uygulamalarında kullanılır [20].

OpenCV, çeşitli görüntü işleme görevlerini gerçekleştirmek için geniş bir araç seti sunar. İşte bazı temel görüntü işleme görevleri ve OpenCV'nin bunları nasıl gerçekleştirdiği açıklanmıştır.

- Görüntü okuma ve yazma: OpenCV, çeşitli formatlarda (JPEG, PNG, BMP) görüntü dosyalarını okuyabilir ve yazabilir. cv2.imread ve cv2.imwrite fonksiyonları, görüntü dosyalarını okuma ve yazma işlemlerini gerçekleştirir.
- Görüntü Dönüştürme:OpenCV, görüntülerin renk uzayları arasında dönüşüm yapabilir. Örneğin, bir görüntüyü BGR formatından gri tonlamalı formata dönüştürmek için cv2.cvtColor fonksiyonu kullanılır.
- Filtreleme ve Geometrik Dönüşümler: OpenCV, çeşitli görüntü filtreleme tekniklerini (Gaussian, Median, Bilateral) ve geometrik dönüşümleri (döndürme, ölçeklendirme, kırpma) destekler. cv2.filter2D, cv2.GaussianBlur ve cv2.resize gibi fonksiyonlar bu işlemleri gerçekleştirir.

OpenCV kullanımının avantajları aşağıda listelenmiştir.

- Geniş Kapsam: OpenCV, görüntü işleme ve bilgisayarla görme alanında geniş bir araç seti sunar. Bu, kullanıcıların çeşitli görevleri tek bir kütüphane ile gerçekleştirmesine olanak tanır.
- Çapraz Platform Desteği: OpenCV, Windows, Linux, macOS ve Android gibi çeşitli platformlarda çalışabilir. Bu, kullanıcıların farklı işletim sistemlerinde projelerini geliştirmesini kolaylaştırır.
- Gerçek Zamanlı İşleme: OpenCV, yüksek performanslı ve gerçek zamanlı görüntü işleme görevlerini gerçekleştirebilir. Bu, özellikle robotik ve otonom sistemlerde önemlidir [20].

OpenCV kullanımının dezavantajları aşağıda listelenmiştir.

- Karmaşıklık: OpenCV'nin geniş özellik yelpazesi, öğrenmesi ve kullanması zor olabilen karmaşık bir kütüphane olmasına yol açabilir. Yeni başlayanlar için eğitimi ve projelerde kullanımı zaman alabilir.
- Dokümantasyon ve Topluluk Desteği: OpenCV'nin dokümantasyonu ve topluluk desteği genellikle kapsamlıdır, ancak bazı ileri seviye özellikler ve özel kullanım durumları için yeterli bilgi bulunamayabilir.

3.6 PIL (Python Imaging Library)

Python Imaging Library (PIL), Python programlama dili için güçlü bir görüntü işleme kütüphanesidir. İlk olarak 1995 yılında Fredrik Lundh tarafından geliştirilmiştir. PIL geniş bir görüntü formatı desteği ve esnek API'si sayesinde çeşitli görüntü işleme görevlerini gerçekleştirmeyi kolaylaştırır. Günümüzde PIL'in gelişimi ara verilmiş olup, onun yerine aynı işlevselliği ve hatta daha fazla özelleği birlikte sunan Pillow adında bir kütüphane kullanılmaktadır [21].

PIL, birçok farklı görüntü formatını okuma, yazma ve dönüştürme kabiliyetine sahiptir. PIL'in kullanıcılar tarafından kullanılmasınaki ana etkenler ve belirli fonksiyonları aşaıda listelenmiştir.

- Görüntü Okuma ve Yazma: JPG,JPEG, PNG, BMP, GIF ve TIFF gibi yaygın görüntü formatlarını destekler.
- Görüntü Dönüştürme: Renk uzayları arasında dönüşüm, yeniden boyutlandırma, kırpma ve döndürme gibi temel görüntü dönüştürme işlemlerini gerçekleştirebilir.
- Filtreleme ve Enhansman: GaussianBlur, Sharpen ve EdgeEnhance gibi çeşitli filtreler uygulayarak görüntüleri iyileştirme.
- Çizim ve Metin Ekme: Görüntülerin üzerine çizim yapma ve metin ekleme kabiliyetine sahiptir.
- Thumbnail Oluşturma: Küçük önizleme görüntüleri oluşturma ve bu görüntülerin optimize edilmesi [22].

PIL, çeşitli görüntü işleme görevleri için kullanılabilir. Başlıca kullanım alanları aşağıda listelenmiştir.

- Web Geliştirme: Web uygulamalarında kullanıcı tarafından yüklenen görüntülerin işlenmesi ve optimize edilmesi.
- Veri Bilimi: Görüntü verilerinin analiz edilmesi ve görselleştirilmesi.
- Makine Öğrenmesi: Görüntü verilerinin ön işleme tabi tutulması ve eğitim veri setlerinin hazırlanması.
- Oyun Geliştirme: 2D oyunlarda sprite'ların ve diğer grafiklerin işlenmesi ve yönetilmesi [23].

PIL kütüphanesinin avantajları aşağıda listelenmiştir:

- Kapsamlı Format Desteği: PIL, birçok yaygın görüntü formatını destekler, bu da onu esnek bir araç haline getirir.
- Basit ve Kullanıcı Dostu API: PIL, kullanıcı dostu ve öğrenmesi kolay bir API sunar, bu da hızlı prototipleme ve geliştirme süreçlerini kolaylaştırır.
- Hız ve Verimlilik: PIL, görüntü işleme görevlerini hızlı ve verimli bir şekilde gerçekleştirir, bu da büyük veri setleri ve gerçek zamanlı uygulamalar için idealdir [22].

PIL kütüphanesinin dezavantajları aşağıda listelenmiştir:

- Sınırlı Güncelleme: PIL'in gelişimi durdurulmuş olup, bazı modern görüntü işleme ihtiyaçlarına cevap veremeyebilir. Bu durum, güncellenmiş ve daha geniş özellik setine sahip olan Pillow kütüphanesinin kullanılmasını gerektirir.
- Kısıtlı Topluluk Desteği: PIL, günümüzde Pillow'un gölgesinde kalmış olup, topluluk desteği ve güncel dökümantasyon açısından sınırlı olabilir [23].

3.7 Tkinter

Tkinter, Python programlama dili için standart bir grafik kullanıcı arayüzü (GUI) kütüphanesidir. Tkinter, Python'un standart kütüphanesinin bir parçası olup, Python ile birlikte gelir ve kullanıcıların masaüstü uygulamaları hızlı ve kolay bir şekilde oluşturmasını sağlar. Tkinter, Tk GUI araç setinin Python bağlamasıdır ve platformlar arası çalışabilirliği sayesinde aynı kodun Windows, macOS ve Linux üzerinde çalışmasını sağlar [24].

Tkinter, kullanıcı dostu ve güçlü bir GUI araç seti sunar. Temel özelliklerinden bazıları şunlardır:

- Kullanıcı Dostu API: Tkinter, öğrenmesi ve kullanması kolay bir API sunar. Bu, yeni başlayanlar için GUI uygulamaları geliştirme sürecini hızlandırır.
- Çapraz Platform Desteği: Tkinter, Windows, macOS ve Linux gibi çeşitli platformlarda çalışabilir. Bu, geliştiricilerin farklı işletim sistemlerinde aynı kodu kullanarak uygulamalarını geliştirmesine olanak tanır.
- Geniş Widget Seçenekleri: Tkinter, düğmeler, etiketler, giriş kutuları, metin kutuları, kaydırma çubukları gibi birçok farklı widget sunar. Bu widget'lar, GUI uygulamalarının temel yapı taşlarını oluşturur.
- Yerel Görünüm ve His: Tkinter, uygulamaların yerel işletim sisteminde doğal görünmesini sağlar. Bu, kullanıcıların uygulamaları kullanırken daha tanıdık bir deneyim yaşamalarını sağlar [24].

Tkinter, birçok farklı uygulama alanında kullanılabilir. Başlıca kullanım alanları şunlardır:

- Eğitim: Tkinter, programlama ve GUI geliştirme öğrenen öğrenciler için mükemmel bir başlangıç noktasıdır. Basit ve anlaşılır API'si sayesinde, öğrenciler hızlı bir şekilde GUI uygulamaları geliştirebilir.
- Hızlı Prototipleme: Tkinter, hızlı prototipleme ve küçük ölçekli projeler için idealdir. Geliştiriciler, fikirlerini hızla görselleştirip test edebilirler.
- Veri Görselleştirme: Tkinter, veri bilimcileri ve analistleri için veri görselleştirme ve basit kullanıcı arayüzleri oluşturmak için kullanılır.
- Masaüstü Uygulamaları: Tkinter, küçük ve orta ölçekli masaüstü uygulamaları geliştirmek için kullanılabilir. Bu uygulamalar, kullanıcıların çeşitli görevleri gerçekleştirmesine olanak tanır [24].

Tkinter kütüphanesinin avantajları aşağıda listelenmiştir.

- Kolay Kullanım: Tkinter, basit ve anlaşılır bir API sunar. Bu, yeni başlayanlar için idealdir ve hızlı bir şekilde GUI uygulamaları geliştirmeyi sağlar.
- Çapraz Platform Desteği: Tkinter, çeşitli platformlarda çalışabilir, bu da geliştiricilerin aynı kod tabanı ile farklı işletim sistemlerinde uygulamalar oluşturmasını sağlar.
- Yerel Görünüm ve His: Tkinter, uygulamaların yerel işletim sisteminde doğal görünmesini sağlar, bu da kullanıcı deneyimini iyileştirir [24].

Tkinter kütüphanesinin dezavantajları aşağıda listelenmiştir.

- Modern Görünüm Eksikliği: Tkinter, bazı kullanıcılar için modern ve çekici bir arayüz sunmaz. Daha modern bir arayüz isteyenler için diğer GUI kütüphaneleri tercih edilebilir.
- Sınırlı Özelleştirme: Tkinter, bazı gelişmiş ve özel GUI gereksinimlerini karşılamayabilir. Bu durumda, daha esnek ve güçlü kütüphaneler kullanılabilir [24].

3.8 NumPy

NumPy (Numerical Python), Python programlama dili için geliştirilmiş, bilimsel hesaplama ve veri işleme alanında yaygın olarak kullanılan bir kütüphanedir. NumPy, büyük, çok boyutlu diziler ve matrisler ile çalışmayı sağlar ve bu veri yapıları üzerinde

hızlı ve verimli hesaplamalar yapmaya olanak tanır. Travis Oliphant tarafından 2005 yılında oluşturulan NumPy, veri bilimciler, mühendisler ve bilim insanları için vazgeçilmez bir araç haline gelmiştir [25].

NumPy, güçlü bir multidimensional array (ndarray) nesnesi ve bu nesneler üzerinde yüksek performanslı operasyonlar sağlayan bir dizi fonksiyon sunar. Temel özelliklerinden bazıları şunlardır:

- Ndarray Nesnesi: NumPy'nin temel veri yapısı olan ndarray, sabit boyutlu ve aynı türdeki öğelerden oluşan çok boyutlu dizileri temsil eder. Bu veri yapısı, hızlı ve verimli hesaplamalar için optimize edilmiştir.
- Matematiksel Fonksiyonlar: NumPy, dizi matematiği için kapsamlı fonksiyonlar sunar. Bu fonksiyonlar arasında aritmetik işlemler, istatistiksel hesaplamalar, lineer cebir işlemleri ve Fourier dönüşümleri bulunur.
- Veri Manipülasyonu ve Şekil Değiştirme: NumPy, dizilerin şeklini değiştirme, birleştirme, bölme ve sıralama gibi veri manipülasyonu görevlerini kolayca gerçekleştirir.
- Random Number Generation (RNG): NumPy, rastgele sayı üretimi ve olasılık dağılımları ile ilgili fonksiyonlar içerir.
- Entegre Kütüphaneler ile Uyumluluk: NumPy, SciPy, pandas, Matplotlib ve scikit-learn gibi diğer bilimsel kütüphanelerle sorunsuz bir şekilde entegre çalışır [25].

NumPy, birçok farklı uygulama alanında kullanılır. Başlıca kullanım alanları şunlardır:

- Veri Bilimi ve Analitiği: NumPy, veri bilimcileri ve analistleri için büyük veri setlerini verimli bir şekilde işlemek ve analiz etmek için temel bir araçtır.
- Makine Öğrenmesi: NumPy, makine öğrenmesi modellerinin geliştirilmesi ve eğitilmesi için temel verileri sağlar. Özellikle, veri ön işleme ve özellik mühendisliği aşamalarında yaygın olarak kullanılır.
- Fizik ve Mühendislik: Fizik ve mühendislik alanlarında, sayısal simülasyonlar ve deneysel veri analizi için kullanılır.
- Finans: Finans sektöründe, portföy optimizasyonu, risk analizi ve zaman serisi analizi gibi görevler için kullanılır [25].

NumPy kütüphanesinin avantajları aşağıda listelenmiştir

- Performans: NumPy, büyük veri setleri üzerinde hızlı ve verimli hesaplamalar yapmayı sağlar. C ve Fortran gibi dillerde yazılmış optimize edilmiş kodlar kullanarak yüksek performans sunar.
- Kapsamlı Fonksiyon Seti: NumPy, geniş bir matematiksel ve istatistiksel fonksiyon seti sunar.
- Kolay Entegrasyon: NumPy, diğer bilimsel kütüphanelerle sorunsuz bir şekilde entegre olur, bu da kapsamlı veri analiz ve modelleme görevlerini kolaylaştırır.
- Kullanıcı Dostu API: NumPy, kullanıcı dostu ve anlaşılır bir API sunar, bu da öğrenmeyi ve kullanmayı kolaylaştırır [25].

NumPy kütüphanesinin dezavantajları aşağıda listelenmiştir.

- Bellek Kullanımı: NumPy, büyük veri setleri ile çalışırken yüksek bellek kullanımı gerektirebilir.
- Öğrenme Eğrisi: Yeni başlayanlar için, özellikle de multidimensional array ve veri manipülasyonu konularında bazı kavramlar zor olabilir.

3.9 Google Colab

Google Colaboratory, yaygın olarak bilinen adıyla Google Colab, Google tarafından geliştirilen ve bulut tabanlı olarak çalışan ücretsiz bir Jupyter notebook hizmetidir. Google Colab, Python kodlarını yazmak, çalıştırmak ve paylaşmak için güçlü bir platform sunar. Veri bilimciler, makine öğrenme uzmanları, araştırmacılar ve öğrenciler tarafından yaygın olarak kullanılan bu platform, özellikle makine öğrenme projeleri ve veri analizi için idealdir [26].

Google Colab, kullanıcıların Python kodlarını herhangi bir yerel kurulum gerektirmeden tarayıcı üzerinden çalıştırmasına olanak tanır. İşte Google Colab'in temel Bulut Tabanlı Çalışma: Google Colab, bulut tabanlı bir platform olduğu için kullanıcılara yüksek hesaplama kaynaklarına erişim sağlar. Bu sayede, kullanıcılar yerel bilgisayarlarının kapasitesi ile sınırlı kalmadan büyük veri setleri üzerinde çalışabilirler.

- Ücretsiz GPU ve TPU Desteği: Google Colab, ücretsiz olarak GPU (Grafik İşlem Birimi) ve TPU (Tensor Processing Unit) desteği sunar. Bu özellik, özellikle derin öğrenme modellerinin eğitimi ve karmaşık hesaplamaların hızlandırılması için büyük bir avantaj sağlar.
- Jupyter Notebook Uyumluluğu: Google Colab, Jupyter notebook formatını destekler. Kullanıcılar, mevcut Jupyter notebook dosyalarını Google Colab'e yükleyebilir ve üzerinde çalışabilirler. Aynı zamanda, Google Colab'de oluşturulan notebook'lar da Jupyter notebook formatında indirilebilir.
- Kolay Paylaşım ve İşbirliği: Google Colab, Google Drive ile entegre çalışır ve kullanıcıların notebook'larını kolayca paylaşmalarına olanak tanır. Bu özellik, ekip çalışmalarında ve eğitim ortamlarında işbirliğini artırır [26].

Google Colab, geniş bir kullanım yelpazesi sunar ve çeşitli alanlarda kullanılabilir:

- Makine Öğrenmesi ve Derin Öğrenme: Google Colab, TensorFlow, Keras, PyTorch gibi makine öğrenmesi kütüphaneleri ile uyumlu çalışır ve bu kütüphaneleri kullanarak modellerin hızlı bir şekilde geliştirilmesini ve eğitilmesini sağlar.
- Veri Analizi ve Görselleştirme: Pandas, NumPy, Matplotlib ve Seaborn gibi veri analizi ve görselleştirme kütüphaneleri ile entegre çalışarak, veri setlerinin analiz edilmesini ve görselleştirilmesini kolaylaştırır.
- Eğitim ve Öğretim: Google Colab, eğitimciler ve öğrenciler için ideal bir platformdur. Ders materyalleri oluşturma, paylaşma ve üzerinde çalışma imkanı sunar. Ayrıca, interaktif kod örnekleri ile öğrenmeyi kolaylaştırır [26].

Google Colab'ın avantajları aşağıda listelenmiştir.

- Kolay Erişim ve Kullanım: Google Colab, herhangi bir yazılım kurulumuna gerek kalmadan tarayıcı üzerinden kullanılabilir. Bu, kullanıcıların hızlı bir şekilde çalışmaya başlamasını sağlar.
- Ücretsiz Hesaplama Kaynakları: Google Colab, ücretsiz olarak GPU ve TPU desteği sunar. Bu, büyük veri setleri ve karmaşık modellerle çalışırken hesaplama süresini önemli ölçüde azaltır.

- Kolay Paylaşım ve İşbirliği: Google Colab, kullanıcıların notebook'larını kolayca paylaşmasına ve işbirliği yapmasına olanak tanır. Bu özellik, ekip çalışmaları ve eğitim ortamları için idealdir [26].

Google Colab'ın dezavantajları aşağıda listelenmiştir.

- Bağlantı ve Süre Sınırlamaları: Google Colab, ücretsiz kullanımda bağlantı ve işlem süresi sınırlamaları getirebilir. Uzun süreli ve yoğun hesaplama gerektiren işlemler için bu sınırlamalar sorun yaratabilir.
- Veri Gizliliği: Google Colab, bulut tabanlı bir hizmet olduğu için veri gizliliği ve güvenliği konusunda endişeler olabilir. Hassas verilerle çalışırken bu durumu göz önünde bulundurmak önemlidir.

3.10 ResNet ve ResNet50

ResNet, yani Derin Artık Ağlar (Deep Residual Networks), derin öğrenme modellerinin daha derin katmanlara sahip olmasını sağlayarak, görüntü sınıflandırma ve nesne tespiti gibi görevlerde önemli başarılar elde eden bir mimaridir. 2015 yılında Kaiming He, Xiangyu Zhang, Shaoqing Ren ve Jian Sun tarafından geliştirilmiş ve 2015 ImageNet yarışmasında büyük başarılar elde etmiştir. ResNet, derin ağların eğitiminde karşılaşılan zorlukları aşmak için yenilikçi bir "artık bağlantı" (residual connection) yapısı kullanır [29].

ResNet'in temel özelliklerinden bazıları aşağıda listelenmiştir.

- Artık Bloklar: ResNet'in en belirgin özelliği, her bir katmanda orijinal girdi ile işlenen çıktıyı birleştiren artık bağlantılardır. Bu yapı, gradyanların kaybolmasını engelleyerek daha derin ağların eğitilmesini mümkün kılar.
- Derinlik: ResNet, 50, 101, 152 katmanlı modeller gibi çok derin yapılandırmalara sahiptir. Bu derinlik, modelin daha karmaşık ve ayrıntılı özellikleri öğrenmesini sağlar.
- Performans: ResNet, hem görüntü sınıflandırma hem de nesne tespiti gibi görevlerde yüksek performans göstermiştir. 2015 ImageNet yarışmasında 152 katmanlı ResNet, %3.57 hata oranı ile birinci olmuştur [29].

Derin öğrenme modelleri, katman sayısı arttıkça daha karmaşık özellikleri öğrenebilirler. Ancak, katman sayısının artması, gradyanların kaybolması ve ağın aşırı uyum yapması gibi sorunları da beraberinde getirir. ResNet, bu sorunları çözmek için "artık bloklar" kullanır. Artık bloklar, her bir katmandaki çıktıyı orijinal girdiye ekleyerek modeli eğitir.

Artık bloklar, aşağıdaki gibi formüle edilir: $y=F(x,\{Wi\})+x$ Burada x girdi, $F(x,\{Wi\})$ ise katmandaki dönüşümdür. Bu yapı, gradyanların doğrudan ağın başlangıcına geri akmasını sağlar, bu da daha derin ağların eğitilmesini mümkün kılar [29].

ResNet mimarisi, belirli sayıda artık blokların bir araya getirilmesiyle oluşur. Örneğin, ResNet-50 modeli aşağıdaki yapıdadır:

- Conv1: 7x7 Konvolüsyon, 64 filtre, Stride 2
- Max Pooling: 3x3 Max Havuzlama, Stride 2
- Conv2_x: 3x3 Konvolüsyon, 64 filtre, 3 adet Artık Blok
- Conv3_x: 3x3 Konvolüsyon, 128 filtre, 4 adet Artık Blok
- Conv4_x: 3x3 Konvolüsyon, 256 filtre, 6 adet Artık Blok
- Conv5_x: 3x3 Konvolüsyon, 512 filtre, 3 adet Artık Blok
- Average Pooling: Global Ortalama Havuzlama
- Fully Connected Layer: 1000 sınıflı tam bağlantı katmanı (ImageNet için) .

Her bir artık blok, iki veya üç katmandan oluşur ve orijinal girdiyi çıktı ile birleştirir. Bu yapı, derin ağların daha etkili bir şekilde öğrenmesini sağlar.

ResNet, farklı derinliklerde çeşitli modeller sunar. Bu modeller arasında ResNet 9, ResNet-9 Enhanced, ResNet-18, ResNet-34, ResNet-50, ResNet-101 ve ResNet-152 gibi varyantlar bulunmaktadır. Her bir model, artan derinliği ile daha karmaşık ve detaylı özellikleri öğrenme kapasitesine sahiptir. En popüler modellerden biri olan ResNet-50, 50 katmanlı bir derin ağ yapısına sahiptir ve genellikle birçok görüntü işleme ve sınıflandırma görevinde kullanılır [29].

ResNet-50, 50 katmandan oluşan derin bir konvolüsyonel sinir ağıdır. Yapısında 48 konvolüsyon katmanı, 1 Max Pooling katmanı ve 1 Average Pooling katmanı bulunur. ResNet-50'nin başlıca özellikleri şunlardır:

- Derinlik ve Detay: 50 katmanlı yapısı, modelin daha karmaşık özellikleri ve desenleri öğrenmesini sağlar. Bu derinlik, özellikle büyük veri setleri ve karmaşık sınıflandırma görevleri için idealdir.
- Verimli Hesaplama: Artık bloklar sayesinde, modelin eğitimi daha verimli ve hızlı bir şekilde gerçekleştirilebilir. Bu yapı, gradyanların kaybolmasını engelleyerek modelin performansını artırır.
- Genel Amaçlı Kullanım: ResNet-50, birçok farklı görüntü işleme ve sınıflandırma görevinde kullanılır. Hem akademik araştırmalarda hem de endüstriyel uygulamalarda yaygın olarak tercih edilir .

ResNet mimarisinin avantajları aşağıda listelenmiştir:

- Daha Derin Ağlar: Artık bloklar sayesinde, ResNet çok daha derin ağların eğitilmesini mümkün kılar. Bu, modelin daha karmaşık özellikleri öğrenmesini sağlar.
- Gradyanların Kaybolmaması: Artık bağlantılar, gradyanların doğrudan ağın başına geri akmasını sağlar, bu da gradyan kaybolma sorununu önler.
- Yüksek Performans: ResNet, birçok görüntü sınıflandırma ve nesne tespiti görevinde yüksek performans göstermiştir. ImageNet gibi yarışmalarda birinci gelmiş ve endüstri standardı haline gelmiştir [29].

ResNet mimarisinin dezavantajları aşağıda listelenmiştir:

- Hesaplama Maliyeti: Çok derin ağlar, büyük hesaplama kaynakları gerektirir. Bu, eğitim süresini uzatabilir ve daha fazla donanım kaynağı gerektirebilir.
- Model Boyutu: Derin modeller, büyük bellek alanı gerektirir. Bu, modelin taşınabilirliğini ve dağıtılabilirliğini zorlaştırabilir.

4. YÖNTEMLER

Bu bölümde, tezin geliştirilmesi aşamasındaki adımlar, projenin implementasyonu ve projeyle ilgili kullanılan işlemler detaylıca kronolojik implemente edilme sırasıyla açıklanmıştır.

4.1 Veri Setinin Oluşturulması

Proje'nin ilk aşaması veri setini oluşturmaktır. Veri setinin kalitesi ve miktarı, projenin düzgün çalışma düzeyini doğrudan ve en çok etkileyen özelliktir. Bu tür projelerde veri setleri için, hazır veri setleri kullanılabilir. Bu veri setleri daha önceden insanlar tarafından çeşitli yerlerden toplanıp kendi label atanan veri setleridir. Bu veri setleri Kaggle, Google Dataset gibi sitelerden alınabilir.

Proje niş bir konuyu ele aldığı için ve tığ işinin çok fazla çeşidi olduğu için hazır bir veri seti bulunamamıştır. Bu durumda veri setinin sıfırdan oluşturulması planlanmıştır. Veri setini en baştan oluşturmak için yapılan ilk aşama en çok kullanılan en sık yapılan modelleri bulmak olmuştur. Araştırmalar sonucu verilerin kalitesi ve sıklığı üzerine aşağıda sıralanan modeller ele alınıştır:

- Ndarray Nesnesi: NumPy'nin temel veri yapısı olan ndarray, sabit boyutlu ve aynı türdeki öğelerden oluşan çok boyutlu dizileri temsil eder. Bu veri yapısı, hızlı ve verimli hesaplamalar için optimize edilmiştir.
- Alpine Tığ Modeli

Şekil 4.1 Alpine Tığ Modeli Görseli

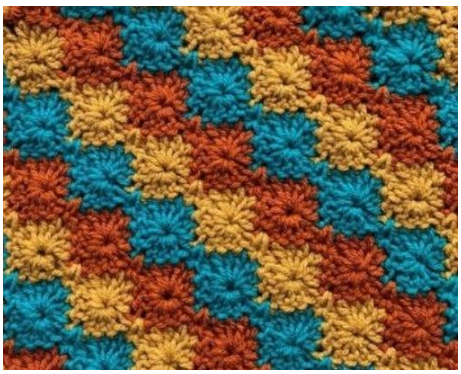
- Basket Weave Tığ Modeli

Şekil 4.2 Basket Weave Tığ Modeli

- Feather Tığ Modeli

Şekil 4.3 Feather Tığ Modeli

- Harlequin Tığ Modeli



Şekil 4.4 Harlequin Tığ Modeli

- Herringbone Tığ Modeli

Şekil 4.5 Herringbone Tığ Modeli

- Jasmine Tığ Modeli

Şekil 4.6 Jasmine Tığ Modeli

- Moss Tığ Modeli

Şekil 4.7 Moss Tığ Modeli

- Shell Tığ Modeli

Şekil 4.8 Shell Tığ Modeli

- Star Tığ Modeli

Şekil 4.9 Star Tığ Modeli

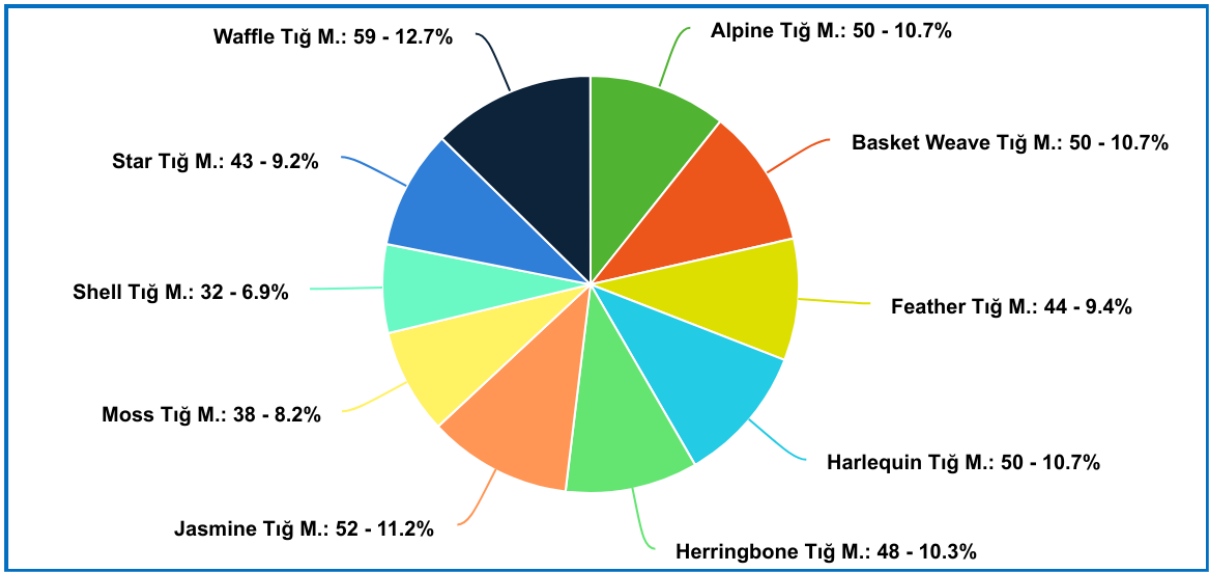
- Waffle Tığ Modeli

Şekil 4.10 Waffle Tığ Model

Tığ modellerine karar verilirken modellerin tekrardan kara ya da dikdörtgen stile uygun, birbiri ardına gelen motifler olmasına dikkat edildi. Ayrıca motiflerin eğitim aşamasında birbirinden ayırt edilebilir olmasına, seçilirken özen gösterildi.

Veriler 30 farklı siteden 10 farklı model olarak toplam 466 adet toplandı. Bu veriler saf verilerdir. Herhangi bir işlem görmemiş verilerdir. Bu verilerin dairesel grafikte dağılım tablosu aşağıda verilmiştir.

Tablo 4.1 Tığ Modeli Dairsel Tablosu



Araştırmanın devamında ise 466 adet verinin bir model eğitimi için yeterli miktarda olmadığı sonucuna varılmıştır. Modelin farklı ve çok veri görmesinin eğitimin önemli bir noktası olması sebebiyle veri artırma (Data Augmentation) yapılması kararı alınmıştır.

4.2 Veri Artırma (Data Augmentation)

Veri artırma, makine öğrenmesi ve derin öğrenme modellerinin performansını artırmak için kullanılan bir tekniktir. Bu teknik, mevcut veri setini çeşitli dönüşümler

uygulayarak genişletmeyi ve çeşitlendirmeyi amaçlar. Veri artırma, özellikle sınırlı veri setlerine sahip projelerde modelin genelleme yeteneğini artırmak ve overfitting (aşırı uyum) sorununu azaltmak için önemli bir rol oynar [27].

Veri artırma, çeşitli yöntemler ve teknikler kullanarak gerçekleştirilebilir. En yaygın kullanılan veri artırma yöntemleri aşağıda listelenmiştir.

- Geometrik Dönüşümler: Döndürme, kaydırma, ölçeklendirme, kırpma ve yansıtma gibi geometrik dönüşümler, görüntülerin farklı perspektiflerden ve ölçeklerde görünmesini sağlar. Bu, modelin verilerdeki varyasyonları daha iyi öğrenmesine yardımcı olur.
- Renk Dönüşümleri: Parlaklık, kontrast, doygunluk ve renk değişiklikleri gibi renk dönüşümleri, görüntülerin farklı ışık koşullarında nasıl göründüğünü simüle eder. Bu, modelin farklı ışıklandırma koşullarında daha iyi performans göstermesini sağlar.
- Gürültü Ekleme: Görüntülere rastgele gürültü eklemek, modelin gürültülü verilerle başa çıkma yeteneğini artırır. Bu teknik, modelin daha sağlam ve dayanıklı olmasını sağlar.
- Kesirsel Maskeler: Görüntünün belirli bölümlerini maskeleyerek veya karartarak, modelin eksik veri ile çalışmasını sağlar. Bu, modelin eksik verileri tahmin etme yeteneğini artırır.
- Senkronize Dönüşümler: Çoklu veri türleri (örneğin, görüntü ve etiket) için senkronize dönüşümler uygulanabilir. Bu, verilerin tutarlılığını ve doğruluğunu korurken, veri setini genişletir [27].

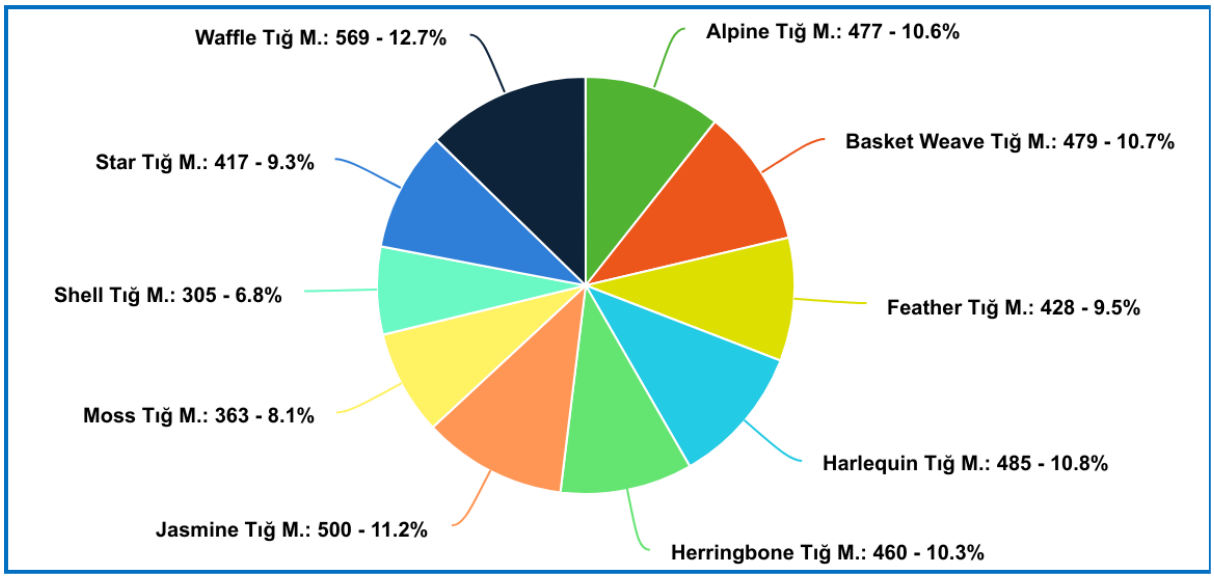
Veri artırma verileri yapay yolla çeşitli işlemlerden geçirdiği için hem avantajları hem de dezavantajları vardır. Veri artırmanın avantajları aşağıda listelenmiştir.

- Genelleme Yeteneği: Veri artırma, modelin genelleme yeteneğini artırır ve overfitting riskini azaltır. Bu, modelin yeni ve görülmemiş verilerde daha iyi performans göstermesini sağlar.
- Veri Setinin Genişletilmesi: Veri artırma, mevcut veri setini genişleterek modelin daha fazla veri ile eğitilmesini sağlar. Bu, özellikle sınırlı veri setlerine sahip projelerde büyük bir avantajdır.

- Modelin Dayanıklılığı: Veri artırma, modelin gürültülü ve eksik verilerle başa çıkma yeteneğini artırır. Bu, modelin daha sağlam ve dayanıklı olmasını sağlar [27].

Bu projede veri artırma model eğitimi için kritik bir rol oynamaktadır. Araştırmalar ve denemeler sonucu veri artırımının hangi yüzdeliklerle gerçekleştiği ve genel bilgiler aşağıda verilmiştir.

Tablo 4.2 Veri Artıma ile Yapılan Tığ Modeli Dairsel Tablosu



Veri artırma uygulamasından sonra Tablo 4.2.1'e ulaşılmıştır. Veri artırımı sonrası toplam artırılmış veri sayısı 4363 olmuştur. Sınıf sınıf veri artırma yüzdeleri aşağıda verilmiştir.

- Alpine Tığ Modelinde saf veri sayısı 50'dir, veri artırma sonrası artırılmış veri sayısı 477'dir. Veri artış oranı ise %854'tür
- Basket Wave Tığ Modelinde saf veri sayısı 50'dir, veri artırma sonrası artırılmış veri sayısı 479'dur. Veri artış oranı ise %858'dir
- Feather Tığ Modelinde saf veri sayısı 44'tür, veri artırma sonrası artırılmış veri sayısı 428'dir. Veri artış oranı ise %872.73'tür
- Harlequin Tığ Modelinde saf veri sayısı 50'dir, veri artırma sonrası artırılmış veri sayısı 485'dir. Veri artış oranı ise %870'dir
- Herringbone Tığ Modelinde saf veri sayısı 48'dir, veri artırma artırılmış sonrası veri sayısı 460'dır. Veri artış oranı ise %858.33'tür

- Jasmine Tığ Modelinde saf veri sayısı 52'dir, veri artırma sonrası artırılmış veri sayısı 500'dir. Veri artış oranı ise %861.54'tür
- Moss Tığ Modelinde saf veri sayısı 38'dir, veri artırma sonrası artırılmış veri sayısı 363'tür. Veri artış oranı ise %855.26'dır.
- Shell Tığ Modelinde saf veri sayısı 32'dir, veri artırma sonrası artırılmış veri sayısı 305'dir. Veri artış oranı ise %853.13'tür
- Star Tığ Modelinde saf veri sayısı 43'dir, veri artırma sonrası artırılmış veri sayısı 417'dir. Veri artış oranı ise %869.77'dir.
- Waffle Tığ Modelinde saf veri sayısı 59'dur, veri artırma sonrası artırılmış veri sayısı 569'dir. Veri artış oranı ise %864.41'dir.

Bu verilere göre ortalama artış oranı %861.72'dir.

4.3 Motif Tespit Etme

Bu projede, tığ işi motiflerinin sınıflandırılması ve tespiti için derin öğrenme modelleri kullanılmıştır.ResNet50 tabanlı bir model eğitilmiş ve bu model kullanılarak tığ işi motiflerinin doğru sınıflandırılması amaçlanmıştır. Aşağıda, bu süreçte izlenen adımlar ve kullanılan yöntemler detaylı bir şekilde açıklanmıştır.

Projedeki önemli noktalardan birisi ise predict fonksiyonudur. Eğitilen modeli kullanarak kullanıcı tarafından yüklenen verinin hangi sınıfa ait olduğunu hesaplayan bir fonksiyon yapılmıştır.

```
def predict(self, file_path):
    img = load_img(file_path, target_size=(224, 224))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)
    predictions = self.model.predict(img_array)
    predicted_class = np.argmax(predictions, axis=1)[0]
    predicted_label = self.class_indices[predicted_class]
    ui.result_label.config(text=f"Assumed type of crochet motif: {predicted_label}")
    self.current_instructions = self.instructions.get_instructions(predicted_label)
    ui.go_button.pack(pady=10)
```

Şekil 4.11 Predict Fonksiyonu Kodu

Bu kodun nasıl işlediği önemi neticesiyle aşağıda sırasıyla açıklanmıştır.

- img = load_img(file_path, target_size=(224, 224))

load_img fonksiyonu, belirtilen dosya yolundan (file_path) bir görüntü yükler ve bu görüntüyü belirtilen boyutlara (224x224 piksel) yeniden boyutlandırır. Bu, modelin girdisinin sabit boyutta olmasını sağlar, çünkü çoğu derin öğrenme modeli sabit boyutlu girdilerle çalışır.

- img_array = img_to_array(img)

img_to_array fonksiyonu, yüklenen ve yeniden boyutlandırılan görüntüyü bir Numpy dizisine dönüştürür. Bu adım, görüntünün modelin işleyebileceği sayısal forma getirilmesini sağlar.

- img_array = np.expand_dims(img_array, axis=0)

np.expand_dims fonksiyonu, görüntü dizisinin boyutunu genişletir. Bu adım, modelin beklediği girdi biçimini elde etmek için gereklidir. Genellikle modeller, bir grup görüntü üzerinde tahmin yapmaya uygun şekilde eğitilir, bu nedenle girdinin ilk boyutu örnek sayısını belirtir (bu durumda tek bir görüntü).

- img_array = preprocess_input(img_array)

preprocess_input fonksiyonu, görüntü dizisini modelin gereksinim duyduğu şekilde ön işler. Bu işlem genellikle, görüntüdeki piksel değerlerinin normalizasyonu ve modelin eğitim sırasında kullanılan aynı veri dönüşümlerinin uygulanmasını içerir.

- predictions = self.model.predict(img_array)

self.model.predict fonksiyonu, ön işlenmiş görüntü üzerinde tahmin yapar. Model, her sınıf için olasılık değerleri döndürür.

```
def __init__(self, model_path):
    self.model = tf.keras.models.load_model(model_path)
    self.class_indices = {'Harlequin Stitch': 0, 'alpine crochet': 1, 'basket weave crochet stitch': 2,
                          'feather stitch': 3, 'herringbone crochet': 4, 'jasmine crochet': 5, 'moss crochet': 6,
                          'shell crochet': 7, 'star stitch': 8, 'waffle stitch': 9}
```

Şekil 4.12 Modelin Eğitim Sınıflarına Verildiği Liste

- predicted_class = np.argmax(predictions, axis=1)[0]

np.argmax fonksiyonu, en yüksek olasılık değerine sahip sınıfın indeksini döndürür. axis=1 parametresi, tahminlerin satırlar boyunca en yüksek değerini alır (örneğin, farklı sınıflar için olasılık değerleri arasından en yüksek olanını seçer).

- predicted_label = self.class_indices[predicted_class]

self.class_indices sözlüğü, sınıf indekslerini karşılık gelen sınıf etiketlerine çevirir. Tahmin edilen sınıfın etiketi, predicted_label değişkenine atanır.

- ui.result_label.config(text=f"Assumed type of crochet motif: {predicted_label}")

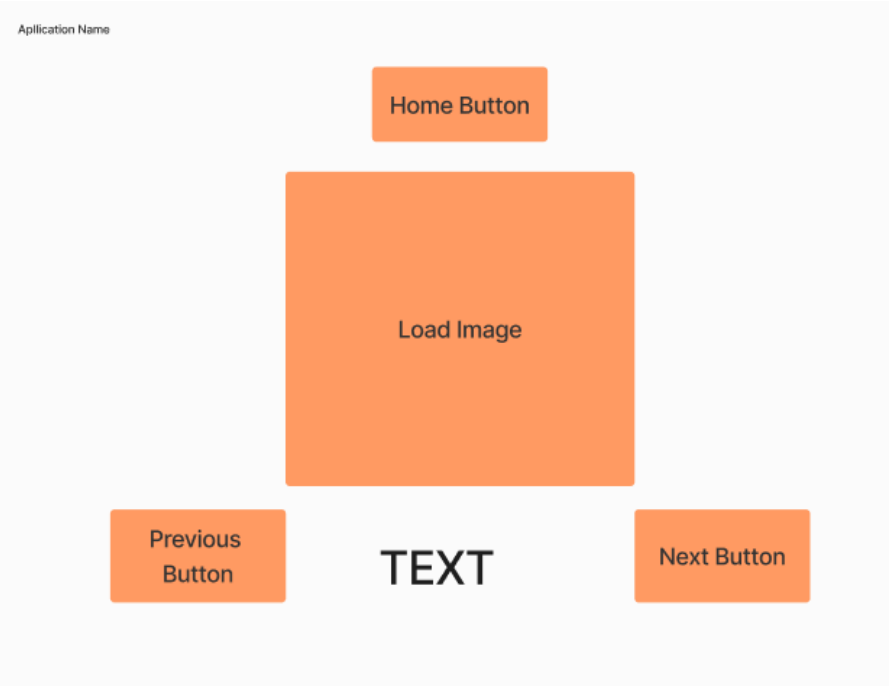
ui.result_label.config fonksiyonu, kullanıcı arayüzünde tahmin edilen motif türünü gösterir. Bu, tahmin sonucunun görsel olarak kullanıcıya sunulmasını sağlar.

- self.current_instructions = self.instructions.get_instructions(predicted_label)
  ui.go_button.pack(pady=10)

self.instructions.get_instructions fonksiyonu, tahmin edilen motif türüne göre talimatları alır. Ardından, "Go to Instructions" butonu kullanıcıya gösterilir, böylece kullanıcı bu talimatlara erişebilir.

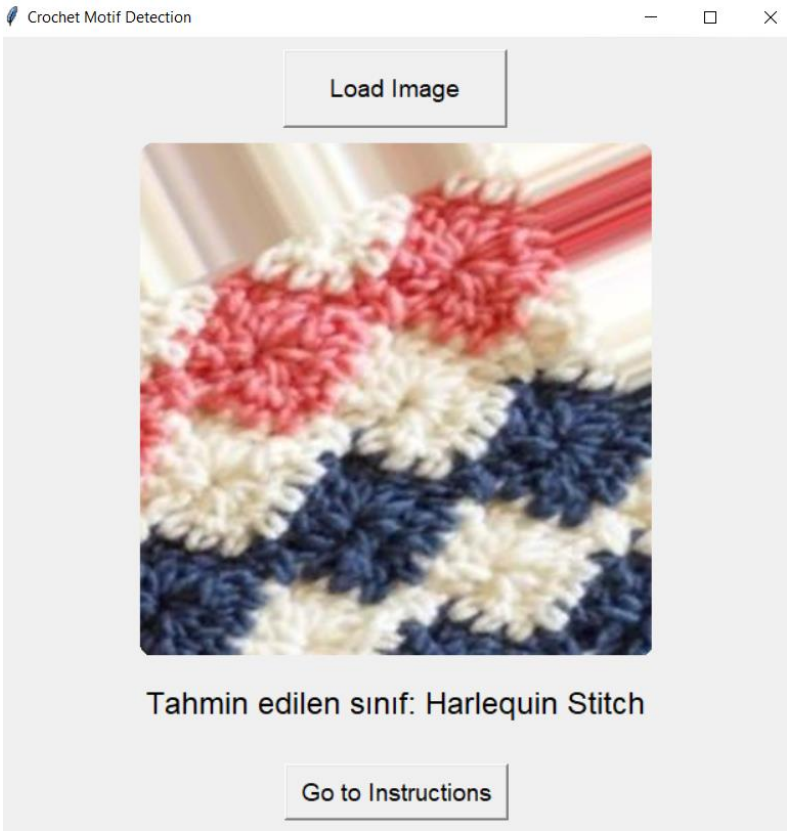
5. ARAYÜZ

Tezde arayüz olabildiğince sade bir biçimde yapılandırılmaya çalışmıştır. Kullanıcıların kolaylıkla anlayabileceği bir ve etkileşebileceği bir uygulama implemente edilmiştir. Arayüzü yapılırken öncelikle ana bir dizayn yapılmıştır. Bu ana dizaynı aşağıda görülebilir.

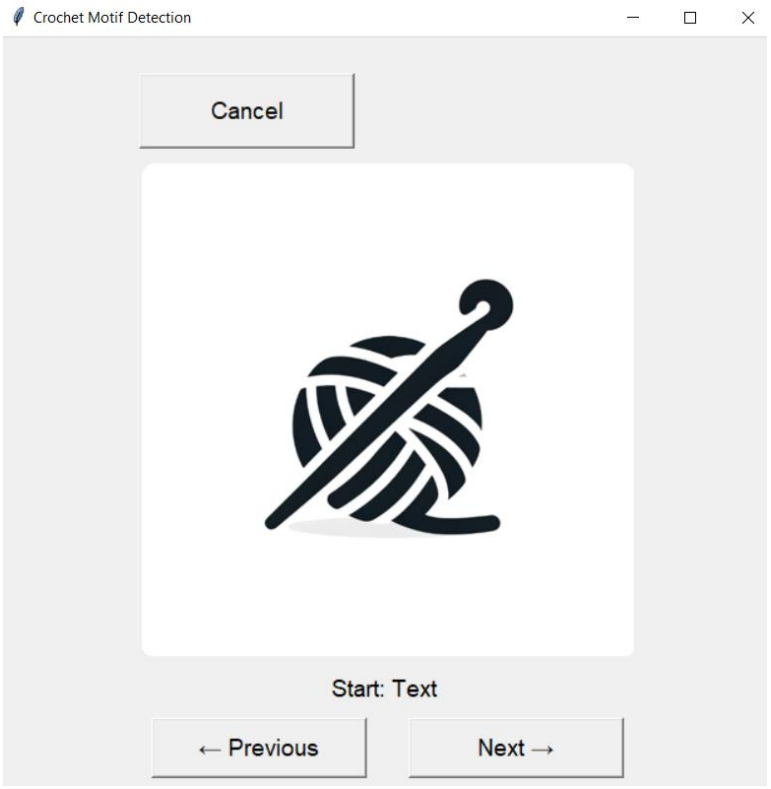


Şekil 5.1 Ana UI Dizaynı

Ana UI' dizaynı hazırlandıktan sonra bu gelişmeler implemente edilmeye başlamıştır.

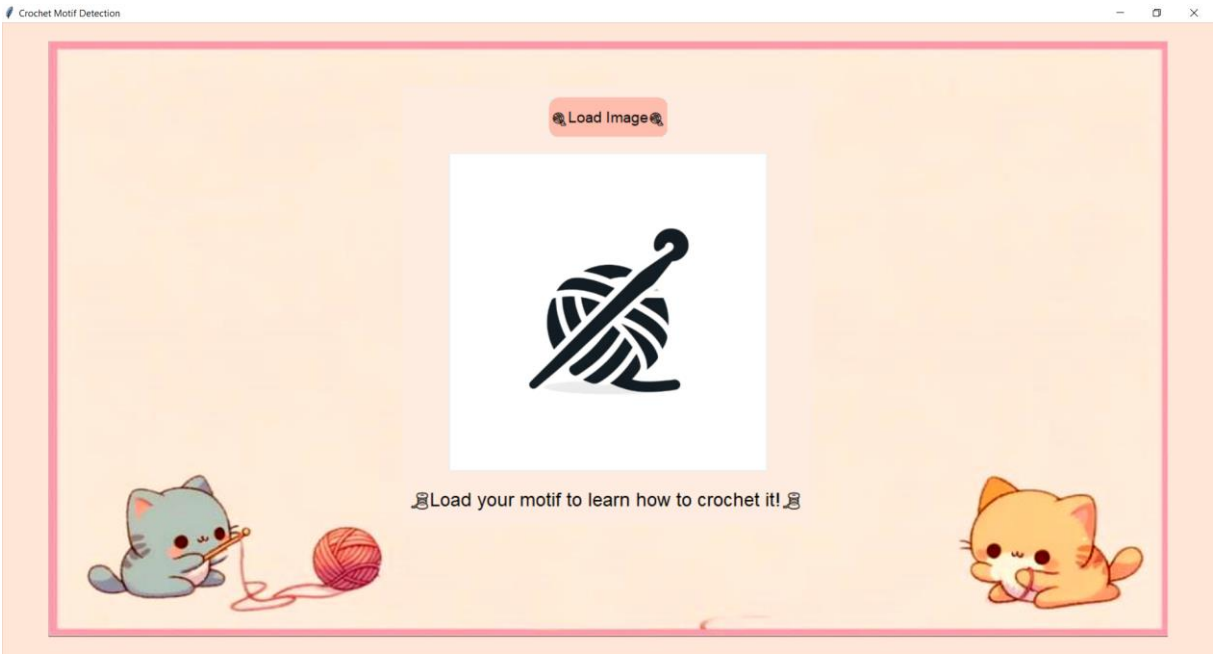


Şekil 5.2 UI Düzenlenmeleri Yapılmadan Geliştirilen Arayüz



Şekil 5.3 UI Düzenlenmeleri Yapılmadan Geliştirilen Arayüz

Arayüz kuralları belirlendikten sonra, kurallar çerçevesinde çalışan uygulama programlanmaya başlamıştır. Belirlenen arayüzde kullanıcının ilk göreceği ekran yükleme ekranı olmaktadır. Görüntü yüklendikten sonra ise motifin yapılma aşamaları gelmektedir. Ana mantık ve çalışma prensibi oluşturulduktan sonra kullanılıcaya sunulacak son hali hazırlanmıştır. Aşağıda tezin arayüzünün son hali gösterilmiştir. Aynı zamanda örnek bir kullanım da gösterilmiştir.



Şekil 5.4 Projenin Ana Sayfası

Proje kullanıcı tarafından ilk açıldığı Şekil 5.3'te görünen ana sayfa kullanıcının karşısına çıkar. Anasayfa anlaşılır sade bir yapıdadır. Sadece tek bir buton bulunur. 'Load Image' butonu kullanıcıya öğrenmek istediği motifi yükleme sekmesi açar.

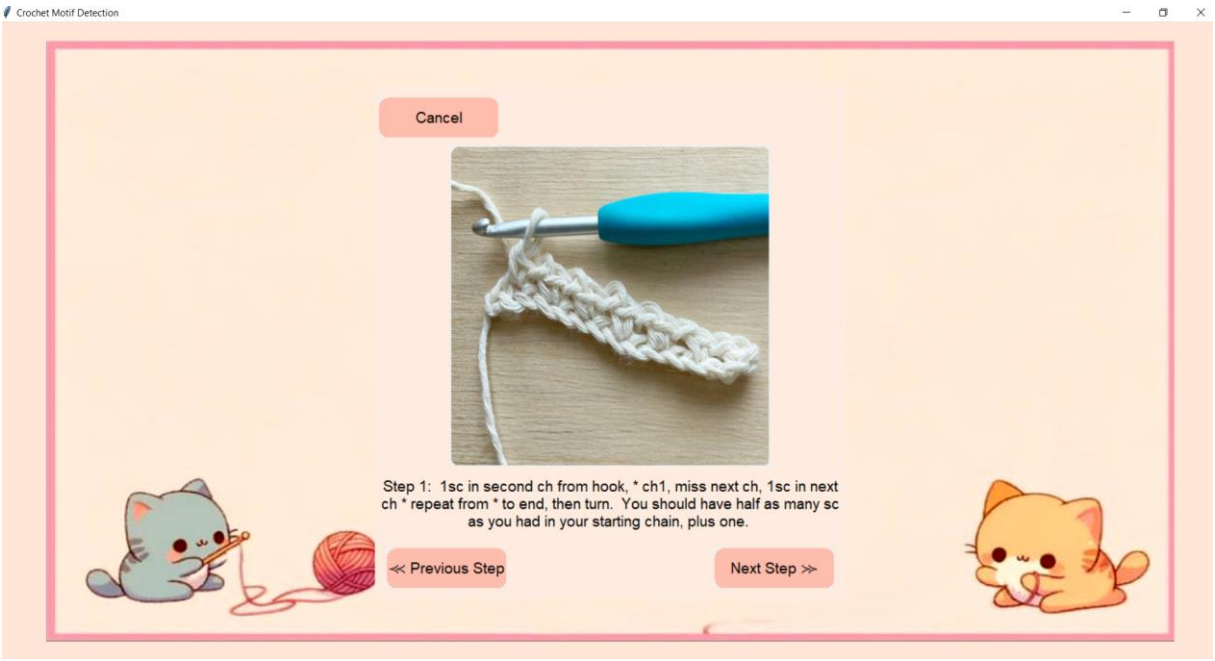
Anasayfanın genel teması yumuşak renkler olarak belirlenmiştir. Renkler birbiriyle uyumlu ve sadedir. Kullacının anlaması için yüklenmesi istenilen yere bir tığ ve çile ilistürasyonu konulmuştur.

Şekil 5.5 İstenilen Resim Yüklendikten Sonra Gelen Ekran

İstenilen resim Şekil 5.4'de görüldüğü üzere yüklendikten sonra ekran değişir. Ekrana hangi motif olduğunu tahmin edilen yazı gelir. Bu yazıya göre 'go to instruction' buton gelir. Bu buton programın tahmin ettiği tığ modelinin nasıl yapıldığını anlattığı sayfaya götürür.

Şekil 5.6 Tığ Modelinin Nasıl Yapıldığını Anlatan Sayfa Örneği

Her modelin anlatım sayfasındaki ortak nokta Şekil 5.5'te de gözüken modelin ana diagramı ve Modelde kullanılan tığ tekniklerinin yazılmasıdır. Bu diagramlar evrensel ve semboliktir. Diagramı okumayı bilen biri anlatıma gerek kalmadan da tığ modelini yapabilir. Buna benzer şekilde modeli anlatımlarla öğrenen bir kişi alıştıktan sonra sadece diagrama ve tığ tekniklerine bakmakta kullanabilir.



Şekil 5.7 Tığ Modelinin Nasıl Yapıldığını Anlatan Sayfa Örneği

Next step butonuna tıklayan bir kullanıcı bundan sonra modelin nasıl yapıldığını anlatan ve görsellerle adım adım gösteren sayfaya gelir. Her modelin yapılışına göre adım sayısı değişmektedir. Adımlar arası dönüş ve geçiş sağlanmaktadır. Cancel butonu ise herhangi bir aşamada anasayfaya dönmek amaçlı kullanılır. Cancel'a basıldığı zaman eklediğiniz son görsel anasayfa kalır ve tekrar yapım aşamasına gidilebilir. Aynı zamanda anasayfaya döndükten sonra istenilen kadar model yüklenip adımlara erişilebilir.

6. DENEYSEL ÇALIŞMALAR

Bu bölümde, tığ işi motiflerinin sınıflandırılması için kullanılan ResNet50 tabanlı derin öğrenme modelinin eğitimi detaylı bir şekilde açıklananmıştır. Veri hazırlama aşaması, modelin başarısı için kritik öneme sahiptir. Bu projede, Google Drive üzerinde saklanan bir veri seti kullanılmıştır. Veri seti, çeşitli tığ işi motiflerinden oluşmaktadır. Her bir motifin yüksek çözünürlüklü görüntüleri, sınıflandırma işlemi için etiketlenmiştir.

Görüntü verileri, modelin girdi boyutlarına uygun hale getirilmiş ve ön işleme fonksiyonları kullanılarak normalleştirilmiştir. Bu süreç, modelin daha iyi performans göstermesi için gereklidir.

Bu projede, önceden eğitilmiş ResNet50 modeli kullanılmış ve üzerine özel katmanlar eklenerek model yapılandırılmıştır. ResNet50 modeli, ImageNet veri seti üzerinde eğitilmiş ağırlıklarla başlatılmış ve son katmanlar özelleştirilmiştir. Model derlendikten sonra, eğitim veri seti üzerinde eğitilmiştir.

Model, belirli bir dönem (epoch) boyunca eğitim veri seti üzerinde eğitilmiş ve doğrulama veri seti kullanılarak performansı değerlendirilmiştir. Eğitim sırasında, her bir dönem sonunda modelin doğrulama performansı izlenmiş ve gerekli iyileştirmeler yapılmıştır.

Tablo 6.1 Modelin Eğitim ve Doğruluk Grafiği

Eğitim Kaybı: Başlangıçta 1.8688 olan eğitim kaybı, her epoch ile düşerek son epoch'ta 0.2139'a kadar indi. Bu, modelin eğitim verilerini zamanla daha iyi öğrendiğini gösterir.

Eğitim Doğruluğu: Başlangıç doğruluğu %42.51 iken, son epoch'ta %93.49'a yükselir. Bu da modelin eğitim verilerindeki örüntüleri giderek daha iyi tanıdığını gösteriyor.

Doğrulama Doğruluğu: Başlangıçta %12.71 olan doğrulama doğruluğu, son epoch'ta %67.08'e ulaşmış.

Sonuç olarak model %70'e yakın bir başarı sağlamıştır. Model eğitim durumundayken çeşitli farklı teknikler denenmiştir. Verilerle test aşamasında diğer sınıflardan daha fazla yanlış algılanan sınıflar çıkarılarak eğitim sağlanmıştır. Bu sonuçlara göre Alpine Sınıfı olmadan eğitilen modelin sonuçları asağıdadır:

Eğitim Kaybı: İlk epoch'ta 3.1091 olarak başlayan eğitim kaybı, her epoch ile düşerek son epoch'ta 1.6606'ya iner. Bu, modelin eğitim verilerini giderek daha iyi öğrendiğini gösterir.

Doğrulama Kaybı: Doğrulama kaybı, olağanüstü yüksek değerler almış. İlk epoch'ta 209687.1875 başlamış ve dalgalanmalarla devam etmiş. Bu, modelin doğrulama verilerinde iyi performans gösteremediğini ve büyük olasılıkla overfitting yaşadığını gösterir.

Eğitim Doğruluğu: İlk epoch'ta %24.35 ile başlayan eğitim doğruluğu, son epoch'ta %49.86'ya ulaşmış. Bu, modelin eğitim verilerinde öğrenme gösterdiğini belirtir ancak ideal bir doğruluk seviyesine ulaşamadığını gösterir.

Sonuç olarak sınıf ve veri kaybının büyük ölçüde zarara uğrattı neticesinde bu model kullanılmamıştır.

7. SONUÇ VE GELECEK ÇALIŞMALAR

7.1 Sonuç

Bu tez çalışmasında, tığ işi motiflerinin sınıflandırılması için derin öğrenme modelleri kullanılmış ve başarılı sonuçlar elde edilmiştir. Özellikle ResNet50 tabanlı model, tığ işi motiflerini doğruluk oranlarıyla sınıflandırmada etkili olmuştur. Modelin eğitim sürecinde veri hazırlama, veri artırma ve model yapılandırma gibi önemli adımlar izlenmiştir. Sonuçlar, modelin genelleme yeteneğinin ve gerçek dünya uygulamalarındaki performansının güçlü olduğunu göstermektedir. Modelin doğruluk oranları ve performans metrikleri, projenin başarılı bir şekilde tamamlandığını ve belirlenen hedeflere ulaşıldığını göstermektedir. Ulaşılan hedefler aşağıda listelenmiştir.

- Doğruluk (Accuracy): Modelin doğruluğu, eğitim ve doğrulama veri setlerinde iyi seviyelerde tutulmuştur. Bu, modelin doğru sınıflandırma yapabilme kapasitesini göstermektedir.
- Kesinlik (Precision) ve Duyarlılık (Recall): Bu metrikler, modelin sınıflandırma performansını detaylandırmaktadır. Yüksek kesinlik, modelin doğru pozitif sonuçlar

üretme yeteneğini, yüksek duyarlılık ise modelin gerçek pozitifleri yakalama yeteneğini yansıtır.

- F1 Skoru: F1 skoru, kesinlik ve duyarlılığın harmonik ortalamasını temsil eder ve genel model performansının bir göstergesidir.

Bu tez çalışması, tığ işi motiflerinin otomatik sınıflandırılması konusunda önemli bir katkı sağlamıştır. Elde edilen sonuçlar, hem akademik alanda hem de endüstriyel uygulamalarda kullanılabilir. Bu tür modeller, tığ işi motiflerinin hızlı ve doğru bir şekilde tanımlanmasına yardımcı olabilir, bu da el sanatları, tekstil ve moda endüstrileri için değerli olmaktadır.

7.2 Gelecek Çalışmalar

Bu tez çalışmasında elde edilen sonuçlar, gelecek çalışmalar için çeşitli öneriler sunmaktadır. Aşağıda, bu alandaki gelecekteki araştırmalar ve iyileştirme önerileri yer almaktadır:

- Model Geliştirmeleri: Mevcut modelin performansını artırmak için daha derin ve karmaşık modellerin (örneğin, ResNet101 veya ResNet152) kullanılması düşünülebilir. Ayrıca, transfer öğrenme teknikleri kullanılarak farklı veri setleriyle modelin yeniden eğitilmesi, modelin genelleme yeteneğini artırabilir.

- Veri Seti Genişletme: Daha büyük ve çeşitli veri setleri kullanarak modelin eğitilmesi, sınıflandırma performansını artırabilir. Özellikle farklı tığ işi motifleri ve desenleri içeren geniş kapsamlı veri setleri oluşturulabilir. Aynı zamanda uygulamaya eklenen her veri veritabanına atılıp veri genişletmesi için uygun olup olmadığı kontrol edildikten sonra düzenli veri artışı sağlanabilir.
- Gerçek Zamanlı Uygulamalar: Modelin gerçek zamanlı uygulamalarda kullanılabilirliği artırılabilir. Bu, mobil uygulamalar veya web tabanlı platformlar aracılığıyla kullanıcıların tığ işi motiflerini anında tanımlamasına olanak tanıyabilir. Sadece dosya eklemekle kalmayıp direkt kamera ile gerçek zamanlı bir şekilde algılama methodu geliştirilebilir.

- Çapraz Doğrulama: Modelin performansını daha iyi değerlendirmek için çapraz doğrulama teknikleri kullanılabilir. Bu, modelin farklı veri setleri üzerinde test edilerek daha genelleştirilebilir sonuçlar elde edilmesini sağlayabilir.
- Ensemble Öğrenme: Birden fazla modelin birlikte kullanılması (ensemble learning) ile sınıflandırma performansı artırılabilir. Bu, farklı modellerin güçlü yönlerini birleştirerek daha doğru tahminler yapılmasını sağlayabilir.
- Farklı Alanlarda Uygulama: Bu çalışma, tığ işi motiflerinin yanı sıra farklı el sanatları, tekstil desenleri veya diğer görsel materyallerin sınıflandırılmasında da uygulanabilir. Bu tür genişletilmiş uygulamalar, modelin kullanım alanını artırabilir ve daha geniş bir kitleye hitap edebilir.
- Uygulama içi etkileşim eklenebilir. Böylece kullanıcı sayısı artırılmaya odaklı bir uygulama haline gelebilir. Online bir platform hali alıp kullanıcılar arası etkileşim, ders gibi çeşitli aktiviteler sağlanabilir.

KAYNAKÇA

1. Mini Crochets, 2024 [Çevrimiçi], https://minacrochets.com/ (Erişim tarihi: 19 Mayıs 2024).

2. Crochet Penguin, 2024 [Çevrimiçi], https://crochetpenguin.com/ (Erişim tarihi: 19 Mayıs 2024).

3. I Like Crochet, 2024 [Çevrimiçi], https://www.ilikecrochet.com/ (Erişim tarihi: 19 Mayıs 2024).

4. Little World of Whimsy, 2024[Çevrimiçi], https://littleworldofwhimsy.com/ (Erişim tarihi: 23 Mayıs 2024).

5. Crochetpedia, 2024 [Çevrimiçi], https://crochetpedia.com/ (Erişim tarihi: 24 Mayıs 2024).

6. Computer Vision and Image Processing: A Beginner's Guide [Çevrimiçi], https://opencv.org/ (Erişim tarihi: 27 Mayıs 2024).

7. GeeksforGeeks, 2024, *Digital Image Processing Basics* [Çevrimiçi], https://www.geeksforgeeks.org/digital-image-processing-basics/ (Erişim tarihi: 27 Mayıs 2024).

8. Simplilearn, 2024, *Image Processing* [Çevrimiçi], https://www.simplilearn.com/image-processing-article (Erişim tarihi: 30 Mayıs 2024).

9. Wikipedia, 2024, *Digital Image Processing* [Çevrimiçi], https://en.wikipedia.org/wiki/Digital_image_processing (Erişim tarihi: 30 Mayıs 2024).

10. Gonzalez, R. C., & Woods, R. E., 2018, *Digital Image Processing* (4th Edition), Pearson.

11. Szeliski, R., 2010, *Computer Vision: Algorithms and Applications*, Springer.

12. Lutz, M., 2013, *Learning Python* (5th Edition), O'Reilly Media. Learning Python on O'Reilly Media.

13. Wikipedia, 2024, *Yapay zekâ* [Çevrimiçi], *https://tr.wikipedia.org/wiki/Yapay_zek%C3%A2* (Erişim tarihi: 1 Haziran 2024).

14. Abadi, M., et al., 2016, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, TensorFlow White Paper.

15. Chollet, F., 2018, *Deep Learning with Python*, Manning Publications. Deep Learning with Python on Manning.

16. TensorFlow Documentation, 2024, *TensorFlow Official Website* [Çevrimiçi], *https://www.tensorflow.org/* (Erişim tarihi: 4 Haziran 2024).

17. Chollet, F., 2018, *Deep Learning with Python*, Manning Publications. Deep Learning with Python on Manning.

18. Brownlee, J., 2019, *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*, Packt Publishing. Deep Learning for Computer Vision on Packt.

19. OpenCV Documentation, 2024, *OpenCV Official Website*[Çevrimiçi], *https://opencv.org/* (Erişim tarihi: 6 Haziran 2024).

20. Kaehler, A., & Bradski, G., 2016, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, O'Reilly Media. Learning OpenCV 3 on O'Reilly Media.

21. Pillow Documentation, 2024, *Pillow Official Documentation* [Çevrimiçi], *https://pillow.readthedocs.io/* (Erişim tarihi: 7 Haziran 2024).

22. Clark, A., 2015, *Pillow (PIL Fork) Documentation*, Pillow Documentation on Read the Docs. Pillow Documentation.

23. Lundh, F., 2009, *Python Imaging Library Handbook*, PIL Handbook on Effbot. PIL Handbook.

24. Tkinter Documentation, 2024, *Tkinter Official Documentation* [Çevrimiçi], *https://docs.python.org/3/library/tkinter.html* (Erişim tarihi 10 Haziran 2024).

25. NumPy Documentation, 2024, *NumPy Official Documentation* [Çevrimiçi], *https://numpy.org/doc/stable/* (Erişim tarihi: 10 Haziran 2024).

26. Google Colab Documentation, 2024, *Google Colab Official Documentation* [Çevrimiçi], *https://colab.research.google.com/notebooks/intro.ipynb* (Erişim tarihi: 11 Haziran 2024).

27. Shorten, C., & Khoshgoftaar, T. M., 2019, *A survey on Image Data Augmentation for Deep Learning*, Journal of Big Data, 6(1), 60. Journal of Big Data.

28. He, K., Zhang, X., Ren, S., & Sun, J., 2016, *Deep Residual Learning for Image Recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). Deep Residual Learning.

29. ResNet Documentation, 2024 [Çevrimiçi], ResNet Official Documentation, https://keras.io/api/applications/resnet/ (Erişim tarihi: 30 Mayıs 2024).