DERİN ÖĞRENME KULLANARAK UÇAK TANIMA

TÜRK HAVA KURUMU ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ZEYNEL ÜNAL

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

AĞUSTOS 2024

Tez Onayı

DERİN ÖĞRENME KULLANARAK UÇAK TANIMA

Türk Hava Kurumu Üniversitesi, Lisansüstü Eğitim Enstitüsü, Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı, Elektrik ve Bilgisayar Mühendisliği (Türkçe) Tezli Yüksek Lisans Programı öğrencisi **Zeynel ÜNAL**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirerek, aşağıda imzaları olan jüri önünde tezini başarı ile sunmuştur.

Doç. Dr. Adnan GÜZEL
Müdür, Lisansüstü Eğitim Enstitüsü _____

Dr. Öğr. Üyesi Aytun ONAY
Anabilim Dalı Başkanı, Elektrik ve Bilgisayar Mühendisliği _____

Dr. Öğr. Üyesi Şadi ŞEHAB
Tez Danışmanı: Türk Hava Kurumu Üniversitesi _____

Tez Jüri Üyeleri :

Doç. Dr. Mustafa KAYA
Havacılık Ve Uzay Mühendisliği Bölümü,
Ankara Yıldırım Beyazıt Üniversitesi _____

Dr. Öğr. Üyesi Hasan AKSOY
Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği
Türk Hava Kurumu Üniversitesi _____

Dr. Öğr. Üyesi Şadi ŞEHAB
Mühendislik Fakültesi, Bilgisayar Mühendisliği
Türk Hava Kurumu Üniversitesi _____

Tarih: 15 Ağustos 2024

Tez yazım kurallarına uygun olarak hazırladığım "Derin Öğrenmeyi Kullanarak Uçak Tanima" konulu tez çalışmasında tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi, görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu, başkalarının eserlerinden yararlanmada ilgili eserlere bilimsel normlar dahilinde atıfta bulunduğumu, atıfta bulunduğum her eserin tümünü kaynak olarak gösterdiğimi, kullanılan verilerde herhangi bir tahrifat yapmadığımı ve bu tezin herhangi bir bölümünü bu üniversite ya da başka bir üniversitede başka bir tez çalışması olarak sunmadığımı beyan ederim.

Zeynel ÜNAL

ABSTRACT

AIRCRAFT RECOGNITION WITH DEEP LEARNING

Unal, Zeynel
Master of Science, Electrical and Computer Engineering
Supervisor : Assist. Prof. Dr. Şadi ŞEHAB

August 2024, 71 pages

Deep learning is a machine learning method that uses artificial neural networks and large datasets to solve complex problems. Aircraft recognition is an application area of deep learning techniques. Deep learning can be used in image processing and detection tasks for aircraft recognition. For example, deep neural networks can be used to automatically recognize aircraft. Such a system can accurately identify aircraft by analyzing their shapes, wing and engine arrangements, aircraft sizes, and other characteristics. Deep learning can process data obtained from radar, infrared, and optical sensors for aircraft recognition. This data can be fed into deep neural networks for tasks such as detecting, classifying, and tracking aircraft. Additionally, deep learning techniques can be used for important tasks like anomaly detection and fault prediction in aircraft systems. Sensor data from aircraft can be analyzed by deep neural networks to distinguish between normal operating conditions and potential issues. This allows for the prediction of potential faults in aircraft, enabling appropriate measures to be taken in advance. Deep learning has various applications in aircraft recognition, often providing significant contributions to the aviation industry in areas such as safety, maintenance, and planning. However, a good dataset and continuously updated algorithms are required for the reliability and accuracy of

these systems. A model proposed for artificial noise reduction using deep learning is Convolutional Neural Networks (CNNs)-based autoencoder models. An autoencoder is a type of artificial neural network that creates an encoding function to represent input data and then reconstructs the original input from this encoding. This type of model can be used to convert noisy images into cleaned (denoised) images. Specifically, a type of autoencoder known as a "Denoising Autoencoder" is trained to convert noisy input data into its cleaned versions. When noisy images are provided to the network, it automatically learns to reduce noise and produce images similar to the original clean data. In collaborative studies on recognizing fighter jets using deep learning models, the model has been further developed to achieve levels up to 99% accuracy.

August 15th, 2024.

Keywords: Deep Learning, Aircraft Recognition, Artificial Intelligence

ÖZ

DERİN ÖĞRENME KULLANARAK UÇAK TANIMA

Ünal, Zeynel
Yüksek Lisans, Elektrik ve Bilgisayar Mühendisliği
Tez Yöneticisi: Dr. Öğr. Üyesi Şadi ŞEHAB

Ağustos 2024, 71 sayfa

Derin öğrenme, yapay sinir ağları ve büyük veri kümeleri kullanarak karmaşık problemleri çözmek için kullanılan bir makine öğrenme yöntemidir. Uçak tanımı da derin öğrenme tekniklerinin bir uygulama alanıdır. Derin öğrenme, uçak tanımı için görüntü işleme ve algılama görevlerinde kullanılabilmektedir. Örneğin, uçakları otomatik olarak tanımak için derin sinir ağları kullanılabilmektedir. Bu tür bir sistem, uçakların şekillerini, kanat ve motor düzenini, uçak boyutlarını ve diğer özellikleri analiz ederek uçakları doğru bir şekilde tanıyabilmektedir. Derin öğrenme, uçak tanımında radar, kızılötesi ve optik sensörlerden elde edilen verileri işleyebilmektedir. Bu veriler, derin sinir ağlarına beslenerek uçakların tespit edilmesi, sınıflandırılması ve izlenmesi gibi görevlerde kullanılabilmektedir. Ayrıca, derin öğrenme teknikleri uçak sistemlerinde anormallik tespiti ve arıza tahmini gibi önemli görevlerde de kullanılabilmektedir. Uçaklardan gelen sensör verileri, derin sinir ağları tarafından analiz edilerek normal çalışma durumları ve olası sorunlar arasındaki farklar tespit edilebilmektedir. Bu sayede uçaklarda “meydana gelebilecek arızalar önceden tahmin edilebilir ve uygun önlemler alınabilmektedir. Derin öğrenme, uçak tanımı konusunda çeşitli uygulamalara sahip olup uygulamalar genellikle havacılık endüstrisinde güvenlik, bakım, planlama ve diğer alanlarda önemli katkılar sağlayabilmektedir. Ancak, bu sistemlerin güvenilirliği ve doğruluğu

için iyi bir veri seti ve sürekli güncellenen algoritmalar gerekmektedir. Derin öğrenme ile yapay gürültü giderme (yapay gürültü azaltma) için önerilen bir model, Convolutional Neural Networks (CNNs) tabanlı autoencoder modelleridir. Autoencoder, giriş verisini temsil etmek için bir kodlama işlevi oluşturan ve ardından bu kodlamayı orijinal girişe yeniden dönüştüren bir tür yapay sinir ağıdır. Gürültülü görüntüleri temizlenmiş (gürültüsüz) görüntülere dönüştürmek için bu tür bir model kullanılabilmektedir.

Özellikle, "Denoising Autoencoder" olarak bilinen bir tür autoencoder, gürültülü giriş verilerini temizlenmiş versiyonlarına dönüştürmek için eğitilmiş olup bu model, ağa gürültülü görüntüler verildiğinde, ağ gürültüyü azaltmak ve orijinal temiz verilere benzer görüntüler üretmek için otomatik olarak öğrenmektedir.

Derin Öğrenme modelleri ile yapılan savaş uçaklarının tanınması hakkında ortak çalışmalarda model daha fazla geliştirilerek %99 seviyelerine getirilmiştir.

Anahtar Kelimeler: Derin Öğrenme, Algoritma, Görüntü İşleme, Uçak Öğrenme, Tanımla

TEŞEKKÜR

Yüksek lisans tez çalışmam boyunca her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Şadi ŞEHAB ’a ve tez çalışması sürecinde bilgi ve tecrübelerini esirgemeyen başta abim Dr. Öğr. Üyesi Muhammet ÜNAL ’a, ablam Doç. Dr. İlkay ÜNAL SANSAR ’a, arkadaşlarım Bayram Gök ’e ve Merve Güllü ’ye teşekkürlerimi sunarım.

Yüksek lisans eğitimim süresince yanımda olan ve bu süreçte beni sürekli destekleyen, çalışmamı en iyi şekilde tamamlayabilmemi sağlayan başta eşim Beyhan ÜNAL ve oğlum Muhammet Emir ÜNAL ’a olmak üzere beni bu zamana kadar büyüterek hiçbir imkânı esirgemeyen sevgili annem Meryem ÜNAL ve babam Sami ÜNAL ‘a teşekkür ediyorum.

İÇİNDEKİLER

TABLO LİSTESİ

TABLOLAR

ŞEKİL LİSTESİ

ŞEKİLLER

KISALTMALAR LİSTESİ

KISALTMALAR

CNN : Convolutional Neural Network

Dense : Dense convolutional network

İHA : İnsansız Hava Aracı

SİHA : Silahlı İnsansız Hava Aracı

GPU : Graphics processing unit

ISR : Intelligence, surveillance and reconnaissance

SGD : Stokastik Gradyan İnişi

ImageNet : Görsel nesne tanıma yazılım araştırmalarında kullanılmak üzere tasarlanmış büyük bir görsel veri tabanıdır.

GANs : Generative Adversarial Networks

Conv2D : 2 boyutlu bir evrişimli sinir ağı katmanı

RMSProp : Root Mean Square Propagation

GİRİŞ

1. BÖLÜM

1.1 Motivasyon ve Problem Tanımı

İnternetin evrimi modern dünyanın en önemli dönüm noktalarından birini oluştururken, yapay zekâ ve derin öğrenme gibi teknolojilerin gelişimiyle birlikte bilgi işleme ve anlama yeteneklerimizde devrim niteliğinde değişiklikler yaşanmıştır. İnternetin ilk günlerinde, bilgiye erişim ve paylaşımın artması insanlığın bilgi çağına girişini simgelemektedir. Ancak, bu bilgiyi anlamak ve kullanmak için gelişmiş algoritmalar ve veri işleme yöntemleri gerekmekteydi.

Derin öğrenme, yapay zekanın bir alt dalı olarak, insan beyninin çalışma prensiplerinden ilham alarak karmaşık veri yapılarını anlamaya ve öğrenmeye dayanır. Bu teknoloji, büyük miktarda veriye dayalı olarak karmaşık desenleri ve ilişkileri tespit edebilme yeteneğiyle ön plana çıkarmaktadır. İnternetin geniş veri havuzları derin öğrenmenin gelişimi için mükemmel bir zemin sağlamaktadır. Büyük teknoloji şirketleri ve araştırma kurumları, derin öğrenme algoritmalarını kullanarak daha akıllı arama motorları, doğal dil işleme sistemleri, görüntü tanıma sistemleri ve daha fazlasını geliştirmeye başlamıştır. Günümüzde, internetin sunduğu veri ve derin öğrenme tekniklerinin birleşimi, birçok alanda devrim niteliğinde yeniliklere yol açmaktadır. Sağlık sektöründen otomotiv endüstrisine, finansal analizden perakende satışına kadar pek çok alanda, yapay zekâ ve derin öğrenme çözümleri kullanılarak verimlilik arttırılmakta, kararlar daha bilinçli ve hızlı bir şekilde alınmaktadır. Bu teknolojilerin ilerlemesiyle, internetin sunduğu veri miktarı ve yapay zekâ algoritmalarının gücü giderek arttırmakta, gelecekte daha da büyük yeniliklere açık olmaktadır.

1.2 Önerilen Yöntem ve Modeller

Yapay zekâ uygulamalarının insansız hava araçları ile etkileşimi son yıllarda ciddi oranda artış göstermektedir. İnsansız hava araçlarının uygulama alanlarının genişlemesi, yeni ihtiyaçların doğmasına sebep olmuştur. Gelişen drone ve yapay zekâ teknolojileri bu modifikasyon sürecini hızlandırmıştır. Yapay zekanın mühendislik disiplinlerindeki artışı yeni metotların doğmasına ve alternatif yöntemlerin oluşmasına olanak sağlamıştır. Görüntü işleme teknikleri bu bağlamda geliştirilmiş alternatif bir yöntemdir. Görüntü işleme yöntemleri ile adım adım görüntünün yakalanması, sayısallaştırılması ve iyileştirilmesi gerçekleştirilir. Mevcut çalışmada insansız bir hava aracına (quadcopter) görüntü işleme tekniklerinin uygulanması, algoritmalarının geliştirilmesi ve insansız hava aracının bu yönteme karşı verdiği tepkiler incelenmiştir.

Derin öğrenme yöntemleri, görüntü işleme alanında çeşitli çalışmalara yol açmış ve birçok yeniliği beraberinde getirmiştir.[1]

Görüntü Sınıflandırmada kullanılan derin sinir ağları, görüntülerdeki nesneleri tanımak ve sınıflandırmak için kullanılmaktadır. Örneğin, ImageNet gibi büyük veri setleri üzerinde eğitilmiş derin öğrenme modelleri, görüntülerdeki nesneleri doğru bir şekilde tanıyabilir ve sınıflandırabilmektedir.[2], [3]

Nesne tespiti, bir görüntüdeki belirli nesnelerin konumunu ve sınıfını tespit etmeyi amaçlamaktadır. Derin öğrenme teknikleri, tek bir görüntüde birden çok nesneyi tespit etmek için kullanılabilir ve özellikle nesne tespiti alanında önemli ilerlemeler sağlanmaktadır.

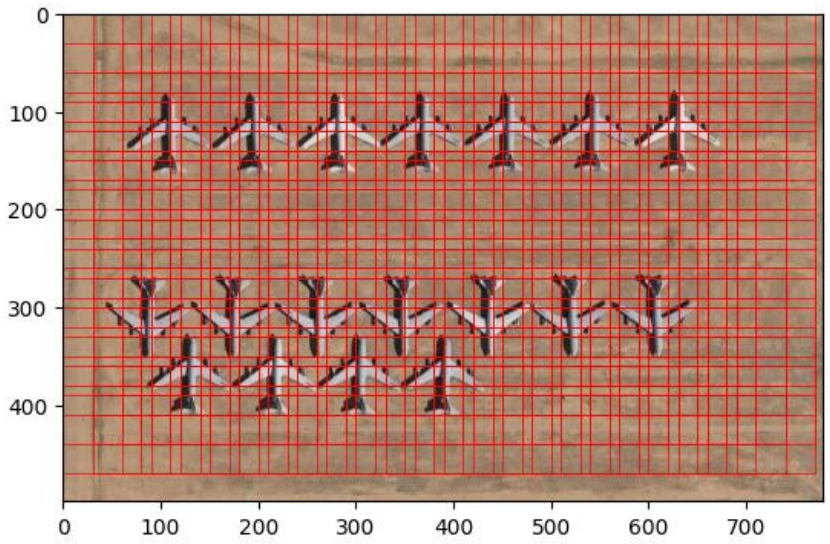
Derin öğrenme, yüz tanıma sistemlerinin geliştirilmesinde önemli bir rol oynamıştır. Büyük veri setleriyle eğitilmiş derin öğrenme modelleri, yüz tanıma sistemlerinin doğruluğunu artırarak gerçek dünya uygulamalarında başarılı sonuçlar vermiştir.

Görüntü üretiminde derin öğrenme, gerçeğe yakın görüntü ve video içeriği üretmek için kullanılabilmektedir. Özellikle Generative Adversarial Networks (GANs) gibi

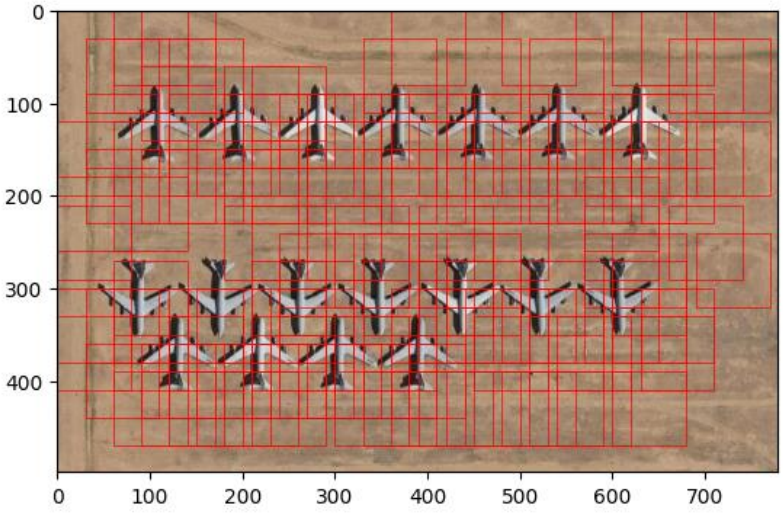
yöntemler, gerçekçi görüntülerin ve videoların oluşturulmasında etkili bir şekilde kullanılmaktadır.[4]

Görüntü segmentasyonu, bir görüntüyü farklı bölgelere ayırma işlemidir. Derin öğrenme teknikleri, piksel seviyesinde segmentasyon için kullanılabilir ve görüntülerdeki nesnelerin sınırlarını ve konturlarını belirlemek için etkili bir şekilde çalışabilmektedir.

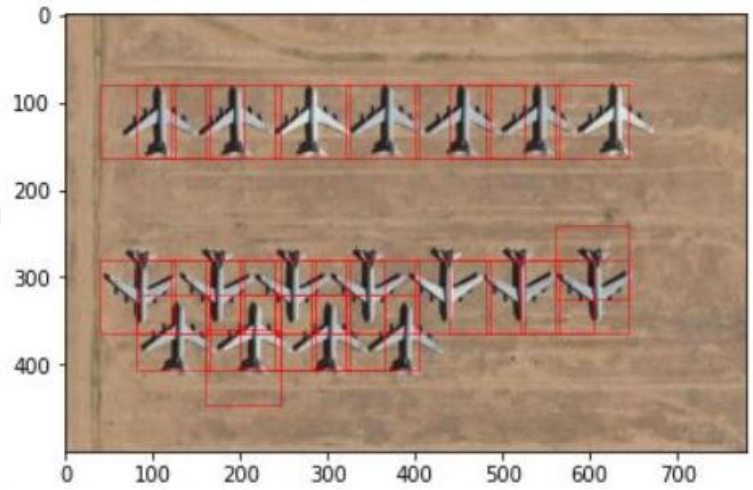
Görüntü restorasyonu ile derin öğrenme, bulanık görüntülerin netleştirilmesi, gürültülü görüntülerin temizlenmesi ve hasarlı görüntülerin onarılması gibi görüntü restorasyonu görevlerinde de kullanılabilmektedir.[5]



Şekil 1.1: Uçakların Tanınabilirliğinin Artırılmasına Yönelik Çalışmalarda Çıkan İlk Sonuçlar



Şekil 1.2: Uçakların Tanınabilirliğinin Artırılmasına Yönelik Çalışmalarda İleri Aşamalar



Şekil 1.3: Uçakların Tanınabilirliğinin Artırılmasına Yönelik Çalışmalarda Son Aşamalar

1.3 Katkılar ve Yenilikler

“Kamuflajı olmayan yerde bilgi gizliliği yoktur.” “Bilginin açığa çıkması bilginin gizliliğine bağlıdır.” Gibi cümlelerden de anlaşılacağı üzere bir yerde bilgi varsa tahmin vardır tahmin varsa doğrulardan yola çıkarak gizliliği açığa çıkarabilecek güce sahip olmaktadır. Burada ifade edilen temel içerik görüntü işleme ile her tür bilginin açığa çıkabileceği anlamını taşımaktadır. Bu da insanları göremediği, algılayamadığı birçok bilginin açığa çıkmasına neden olmaktadır. [6]

Daha öncesinde değerlendirilerek Google Earth üzerinden alınmış olan savaş uçaklarının tür ve cinslerini ayırt edebilmek ve bu verilerin daha kullanılabilir hale dönüştürülmesi hedeflenmiştir. Bu hedef ile birlikte yapay zekâ ile oluşturulan eski modeller hedef alınarak, önceki modeller örnek teşkil ederek daha fazla iyileştirme çalışmaları yapılarak yeni bir modeller oluşturulmaya çalışılmış ve bu verinin doğrulunun daha fazla artırılması hedef alınmıştır. Yeni denemeler sonucunda yeni daha başarılı bir model ortaya çıkmıştır.

2. BÖLÜM

LİTERATÜR ARAŞTIRMALARI

Literatür araştırmalarında yapılan en iyi çalışmalar bu yönde yapılmaktadır. Bu konuda yapılan çalışmalarda derin öğrenme için doğru bilgiyi minimum düzeyde yansıtmak, çalışmak ve anında bu bilgiye ulaşabilmek için çeşitli algoritmalar kullanarak doğru bilgiye ulaşılması için bu yönde yöntemler oluşturulmaktadır. Yapılan araştırmalar sonucunda bazı literatür araştırmalarında en iyi sonuçların doğru sonuçlara ulaşabilmek için derin öğrenmenin yapay zekâ bağlamında daha güçlendirilerek gerçekleştirildiği kanıksanmıştır. [7]

2.1 Araştırma Gereksinimleri

Derin öğrenme ile yapay zeka, özellikle Convolutional Neural Networks (CNNs) kullanarak imaj sınıflandırma alanında büyük başarılar elde edilmektedir. CNN'ler, özellikle bilgisayarlı görü ve görüntü tanıma görevlerinde etkili olan özel bir tür derin öğrenme modelidir. [8]

Evrişim Katmanları (Convolutional Layers), CNN'lerde evrişim katmanlarını kullanarak girdi görüntüden özellik haritası çıkarmaktadırlar. Bu katmanlar, görüntü üzerinde küçük filtrelerin (kernel) belirli bir adımda (stride) kaydırılmasıyla oluşmaktadır. Bu sayede farklı özellikler (kenarlar, köşeler, desenler vb.) çıkarılmaktadır.

Aktivasyon Fonksiyonları, her evrişim katmanının ardından bir aktivasyon fonksiyonu (genellikle ReLU) uygulanarak, özellik haritası üzerindeki negatif değerler düzeltilmektedir.

Pooling (Havuzlama) Katmanları, özellik haritasını örneklemek ve boyutunu küçültmek için kullanılmaktadır. Maksimum havuzlama veya ortalama havuzlama gibi yöntemlerle bir bölgenin en önemli özellikleri korunmaktadır.

Model eğitiminde ve çalışmalarda kullanılan bilgisayar Amd Ryzen 5 6600H – 16 GB ram ve NVidia RTX3050 Ti 4GB olarak belirlenmiştir. Daha sonrasında donamınsal ve yazılımsak karşılaşılan hatalardan dolayı Google Colab kullanılmasına karar verilmiştir.

Tam Bağlantılı Katmanlar (Fully Connected Layers), tam bağlantılı katmanlarda düzleştirilir ve bu katmanlar, sınıflandırma için kullanılmaktadır.

CNN İmaj Sınıflandırma Aşamalarını açıklamak gerekirse;

Veri Toplama ve Ön İşleme olarak kullanılacak veriyi toplamak ve bu veriyi ön işlemektedir. Veri seti, sınıflandırma yapmak istediğiniz nesnelerin fotoğraflarını içermektedir. Veri seti, eğitim ve test olarak iki kısma ayrılmaktadır. Sonrasında CNN modelini oluşturulmaktadır. Model, evrişim katmanları, aktivasyon fonksiyonları, havuzlama katmanları ve tam bağlantılı katmanları içermelidir. Python'da Keras veya TensorFlow gibi derin öğrenme kütüphaneleri kullanarak bir CNN modeli oluşturulabilmektedir. Oluşturulan modeli eğitmek için eğitim veri setini kullanılmaktadır. Bu aşamada model, veri setindeki özellikleri ve sınıfları öğrenmektedir. Eğitim sırasında modelin doğruluğu artırmaktadır. Modeli test veri seti üzerinde değerlendirilmektedir. Modelin doğruluğunu, hassasiyetini ve diğer performans ölçütlerini değerlendirerek modelin başarısını kontrol edilmelidir. Eğitilmiş modeli kullanarak yeni görüntüler üzerinde sınıflandırma tahminleri yapmaktadır. Model, yeni görüntülerdeki nesneleri tanımlamak için öğrendiklerini kullanmaktadır.

Bu örnek, Keras kütüphanesi kullanılarak basit bir CNN modeli oluşturmayı ve eğitmeyi göstermektedir. Veri setinizin, eğitim_seti ve test_seti olarak adlandırılan iki klasöre ayrıldığını unutmayın. Bu klasörler, sınıflandırmak istediğiniz nesnelerin görüntülerini içermektedir.

2.1.1 Veri Setleri ve Ayarlamaları

Google Earth, Google Haritalar, IHA, SIHA ve benzeri insansız hava araçlarından alınmış görüntülerden aluşmaktadır. Savaş Uçağı örnekleri ve yeryüzü örnekleri ile birlikte toplamda 10 adet Görüntü tipi yer almaktadır. Toplam resim sayısı yaklaşık 6300 civarındadır. Belirtilen görüntülerden faydalanılarak bazı belirli savaş uçağı tiplerinden yararlanılmıştır. Bunlar; B-1, B-2, B-52, C-5, C-17, C-130, C-135, E-5, KC-10 ve Yeryüzü görüntülerinden oluşmaktadır.

3. BÖLÜM

3.1 Derin Öğrenme Metotları ve Algoritmaları

Derin öğrenme modellerinde, özellikle Evrişimli Sinir Ağları (CNN) gibi karmaşık yapıdaki ağlarda, optimizasyon önemli bir rol oynamaktadır. Optimizasyon algoritmaları, modelin eğitilmesi sırasında parametrelerin güncellenmesini yönetir ve modelin belirli bir görevi en iyi şekilde öğrenmesini sağlamaya çalışmaktadır. Derin öğrenme ile yapay zekada CNN optimizasyonları aşağıdaki gibi sıralayabiliriz.

3.1.1 FTRL (Follow The Regularized Leader)

FTRL, online öğrenme (veriler parça parça gelir) durumları için özellikle etkili olan bir optimizasyon algoritmasıdır.

3.1.2 RMSProp (Root Mean Square Propagation)

RMSProp, AdaGrad'in öğrenme hızı azalmasını düzeltmek için geliştirilmiştir. Gradyanların hareketli kare ortalamasını almaktadır. Bu sayede daha dengeli bir öğrenme süreci sağlamaktadır.

3.1.3 AdaDelta

AdaDelta, AdaGrad'in öğrenme hızı düşme sorununu çözmek için tasarlanmıştır. Sabit bir öğrenme hızı kullanmamaktadır. Ancak güncellenmiş bir öğrenme hızı kullanmaktadır.

3.1.4 AdaGrad:

AdaGrad, her parametre için özelleştirilmiş bir öğrenme hızı kullanarak eğitimi gerçekleştirmektedir. Nadir görülen özelliklere daha fazla ağırlık verilmesine yardımcı olabilmekle birlikte, eğitim süreci boyunca öğrenme hızı çok düşebilmektedir.

3.1.5 Momentum

Momentum, SGD'ye bir iyileştirme getirmektedir. Gradyanların hareketli ortalamasını kullanarak eğimdeki dalgalanmaları azaltmakta ve bu sayede daha hızlı ve düzgün bir şekilde yakınsak olabilmektedir.

3.1.6 Adam (Adaptive Moment Estimation)

Adam, hem momentum hem de RMSProp'un avantajlarını birleştirmektedir. Adam, eğitim süreci boyunca öğrenme hızını ayarlayarak ve gradyanların ikinci momentini izleyerek oldukça etkili bir optimizasyon sağlamaktadır.

3.1.7 Nadam

Nadam, Nesterov Momentum ile Adam'ın bir kombinasyonudur. Bu kombinasyon, momentum kullanarak gradyanlarda hareket ederken aynı zamanda Adam'ın avantajlarından yararlanmaktadır.

3.1.8 Stokastik Gradyan İnişi (SGD)

SGD, her eğitim örneği için ağı güncellemenin bir formülünü kullanmaktadır. Bu, modelin hızlı bir şekilde eğitilmesine olanak tanımakla birlikte bazı durumlarda dalgalanmalara ve yakınsak sorunlarına neden olabilmektedir.

Bu optimizasyon algoritmaları, modelin hızlı ve etkili bir şekilde eğitilmesini sağlamak için kullanılmaktadır. Hangi optimizasyon algoritmasının kullanılacağına, modelin karmaşıklığına, veri setinin özelliklerine ve eğitim sürecindeki performansa bağlı olarak değişebilmektedir. Çoğu zaman denemelerde ve modelin performansını izlemek için doğrulama verileri kullanılarak en uygun optimizasyon stratejisinin belirlenmesi önerilmektedir.[9]

3.2 FTRL (Follow The Regularized Leader)

FTRL (Follow The Regularized Leader), derin öğrenme modellerinin eğitiminde kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, özellikle büyük ölçekli ve seyrek özellikli veri setlerinde etkili olabilmektedir. FTRL, ağırlık güncellemelerini gerçekleştirirken L1 (Lasso) ve L2 (Ridge) düzenlemelerini birleştirmektedir.

FTRL, her bir özellik için ayrı ağırlıklar ve iki adet yardımcı parametre kullanılmaktadır.

N_i , özellik *i* için gözlem sayısı ve z_i, özellik *i* için gradyan birleştirilmiş etkisidir. Güncelleme kurallarını aşağıdaki gibi gösterebiliriz.

$$z_i = z_i + \text{gradyan} - \text{sign}(z_i) \times \text{L1\_regularization}$$
$$\text{sigma} = \frac{1}{\text{learning rate}} \left(\sqrt{z_i^2 + \text{L2\_regularization}} - \sqrt{\text{L2\_regularization}} \right)$$
$$\text{weight} = -\frac{1}{\text{learning rate} + \frac{\text{beta2} + \sqrt{N_i}}{\text{beta1}}} \times \text{sigma}$$

Bu formülde,

- gradyan, model parametrelerinin gradyanını temsil etmektedir.
- L1_regularization ve L2_regularization, sırasıyla L1 ve L2 düzenlemelerini kontrol eden hiper parametrelerdir.
- learning rate, öğrenme hızını temsil etmektedir.
- beta1 ve beta2, algoritmanın momentum terimleridir.

Büyük veri setleriyle uyumlu olarak FTRL, büyük ölçekli ve seyrek veri setleri üzerinde etkili olabilmektedir. L1 ve L2 düzenlemelerini birleştiren FTRL, L1 ve L2 düzenlemelerini birleştirerek sparsite ve ağırlık büyüklüğü kontrolünü aynı anda sağlamaktadır.

Hiper parametre hassasiyeti olan FTRL, birkaç hiper parametre içermektedir. Bu hiper parametrelerin uygun bir şekilde seçilmesi gerekmektedir. Yüksek hesaplama maliyeti olan FTRL, hesaplama maliyeti yüksek bir algoritmadır. Bu nedenle küçük ölçekli veri setlerinde genellikle tercih edilmemektedir.

FTRL, genellikle büyük ölçekli ve seyrek veri setleriyle çalışırken kullanılmaktadır. Hiper parametrelerin doğru bir şekilde seçilmesi ve yüksek hesaplama maliyeti nedeniyle küçük veri setlerinde tercih edilmeyebilmektedir.[10]

3.3 RMSProp (Root Mean Square Propagation):

RMSProp, özellikle derin öğrenme modellerinde yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, eğitim sürecinde öğrenme hızını düzenleyerek daha hızlı ve daha etkili bir yakınsak sağlamaya çalışmaktadır.

RMSProp algoritması, her parametre için özelleştirilmiş bir öğrenme hızı kullanarak çalışmaktadır. Temel olarak algoritma, her güncelleme adımında gradiyanların karesinin hareketli ortalamasını almakta ve bu kare kökünü alarak bir öğrenme hızı skalası oluşturmaktadır. Bu modelin her parametresi için ayrı bir öğrenme hızı kullanılmasını sağlamaktadır.

RMSProp'ın güncelleme kuralını aşağıdaki gibi gösterebiliriz.

$$\text{cache} = \beta \times \text{cache} + (1 - \beta) \times \text{gradiyan}^2$$
$$\text{parametre} = \text{parametre} - \frac{\text{learning rate} \times \text{gradiyan}}{\sqrt{\text{cache} + \epsilon}}$$

Bu formülde,

- cache, karesel ortalamasıdır.
- β (genellikle 0.9 gibi bir değer), hareketli ortalama faktörüdür.
- gradiyan, mevcut gradyan değeridir.
- learning rate, öğrenme hızını temsil etmektedir.
- ϵ, sıfıra bölme hatasını önlemek için küçük bir sabittir.

Adaptif Öğrenme Hızı ile RMSProp, her parametre için öğrenme hızını adaptif olarak ayarlamaktadır. Bu sayede her parametre için uygun bir öğrenme hızı seçilmesine yardımcı olabilmektedir. Daha hızlı yakınsaklarıyla RMSProp, özellikle non-stationary veri setlerinde ve öğrenme hızını düzenleyerek daha hızlı bir yakınsakları elde etmeye yardımcı olabilmektedir. Öğrenme sürecindeki dalgalanmaları azaltabilmekte ve bu sayede daha stabil bir eğitim süreci sağlayabilmektedir.

Hiper parametre seçiminde RMSProp'ın performansı, kullanılan hiper parametre değerlerine bağlıdır. Bu değerlerin seçimi, deneme-yanılma yöntemleri veya otomatik hiper parametre ayarlama teknikleriyle belirlenebilmektedir. Yüksek boyutlu parametre uzaylarındaki modellerde, bazen performansı düşük olabilmektedir. Bazı durumlarda da, başarılı bir şekilde çalışabilmesi için hassas hiper parametre ayarlarına ihtiyaç duymaktadır.

RMSProp, özellikle derin sinir ağları gibi büyük ve karmaşık modellerde etkili olabilen bir optimizasyon algoritmasıdır. Ancak, model ve veri seti özelliklerine bağlı olarak diğer optimizasyon yöntemleri de tercih edilebilir.[11]

3.4 Tanh (tanjant hiperbolik) aktivasyon fonksiyonu

Tanh (tanjant hiperbolik) aktivasyon fonksiyonu, sigmoid fonksiyonuna benzer bir şekilde çalışarak, çıkış aralığını [-1, 1] arasına genişleten bir aktivasyon

fonksiyonudur. Genellikle derin öğrenme modellerinde ve özellikle Recurrent Neural Networks (RNN'ler) gibi özel durumlarda kullanılmaktadır.

Tanh fonksiyonu matematiksel olarak aşağıdaki gibi gösterebiliriz.

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

Bu formülde,

- $\tanh(x)$, tanh fonksiyonunu temsil etmektedir.
- e, Euler sayısı (yaklaşık olarak 2.71828).
- x, girdi değeridir.

Sınırlı Çıkış Aralığı ile Tanh fonksiyonu, girdiyi -1 ile 1 arasında bir değere dönüştürmektedir. Sigmoid fonksiyonu gibi sınırlı bir çıkış aralığına sahiptir, ancak bu aralık [-1, 1] olarak genişletilmiştir. Asimetrik çıkış ile Tanh fonksiyonu, giriş negatif olduğunda -1'e yaklaşırken, pozitif olduğunda 1'e yaklaşmaktadır. Bu girişin asimetrik etkisini vurgulamaktadır. Tanh fonksiyonunun çıkışının ortalaması sıfırdır. Bu özellik ile sinir ağlarının daha hızlı öğrenmesine yardımcı olabilmektedir. Düz Alan Duyarsızlığı (Vanishing Gradient) ile Sigmoid fonksiyonunda da olduğu gibi Tanh fonksiyonunun türevi de bazı durumlarda çok küçük olabilmektedir. Bu da geriye doğru yayılım sırasında gradiyan kaybına neden olabilmektedir.

Tanh aktivasyon fonksiyonu, genellikle orta katmanlardaki sinir hücrelerinde kullanılmaktadır. Ancak, günümüzde Rectified Linear Unit (ReLU) gibi diğer aktivasyon fonksiyonları, tanh'ın yerine tercih edilmektedir. Çünkü ReLU'nun gradiyan kaybı sorunlarına karşı daha dirençli olduğu ve daha hızlı öğrenme sağladığı gözlemlenmiştir. [12]

3.5 Leaky ReLU (Rectified Linear Unit)

ReLU'nun bir türüdür. Negatif girişler için küçük bir eğimle devam eden bir lineer fonksiyondur. ReLU'nun "dying ReLU" problemine karşı bir çözüm sunmaktadır. Yani negatif girişler nedeniyle bazı nöronlar eğitim sırasında hiç aktive olmamaktadır.

Leaky ReLU fonksiyonu matematiksel olarak aşağıdaki gibi gösterebiliriz.

$$f(x) = \begin{cases} 0.01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} latexlatex f'(x) = \begin{cases} 0.01 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

$$x, \qquad textif x > 0$$
$$alphax, \quad textotherwise\ endcases\ ]$$

Bu formülde,

- $(\ textLeakyReLU(x)\ ),$ Leaky ReLU fonksiyonunu temsil etmektedir.
- $(x\ ),$ girdi değeridir.
- $(\alpha\ ),$ küçük bir pozitif sabittir (genellikle 0.01 veya daha küçük bir değer).

Dying ReLU sorununu azaltmasıyla, negatif girişler için sıfır yerine $(x\ ),$ çıkışını üretmektedir. Bu da negatif girişlerin nöronları "öldürmesini" engellemektedir. Vanishing Gradient sorununu azaltmasıyla da, negatif girişlerde sıfır yerine bir eğimle devam ettiği için geriye doğru yayılım sırasında gradiyan kaybını azaltabilmektedir. Sparsity Teşviki ile, negatif girişlerin bir kısmını sıfıra düşürdüğü için, aktivasyon haritasını seyrek hale getirebilmektedir.

Leaky ReLU, genellikle derin öğrenme modellerinde kullanılan başka bir aktivasyon fonksiyonu olarak tercih edilmektedir. Bununla birlikte, modelinize en iyi uyan aktivasyon fonksiyonunu belirlemek, uyguladığınız probleme ve veri setine bağlı olmaktadır. Bu nedenle farklı aktivasyon fonksiyonlarını denemek faydalı olabilmektedir. [13]

3.6 SELU (Scaled Exponential Linear Unit)

SELU (Scaled Exponential Linear Unit), derin öğrenme modellerinde kullanılan bir aktivasyon fonksiyonudur. SELU, özellikle derin sinir ağlarında kullanıldığında özyineleme (self-normalization) özelliği göstermektedir. Bu ağın daha istikrarlı bir şekilde eğitilmesine yardımcı olabilmektedir.

SELU fonksiyonu matematiksel aşağıdaki gibidir.

$$\begin{matrix} x, & \text{if } x > 0 \\ \alpha \times (e^x - 1), & \text{otherwise } endcases \end{matrix}]$$

Bu formülde,

- $(\text{SELU}(x))$, SELU fonksiyonunu temsil etmektedir.
- (x), girdi değeridir.
- (λ), pozitif bir skalalama sabiti (genellikle 1.0507).
- (α), negatif girişler için küçük bir pozitif sabit (genellikle 1.67326).

Özyineleme (Self-Normalization) ile SELU, ortalaması 0 ve standart sapması 1 olan bir giriş verildiğinde ağın katmanlarını otomatik olarak normalize etme özelliğine sahiptir. Bu ağı daha istikrarlı hale getirerek eğitim performansını artırabilmektedir.

Negatif girişlere duyarlılık ile ELU'ya benzer şekilde, SELU da negatif girişlere karşı sıfırdan farklı bir eğri kullanılmaktadır. Bu da negatif girişlere karşı daha duyarlı bir davranış sergilemektedir. Satürasyon önleme ile SELU'nun eğrisi, negatif girişler için yaklaşık olarak sıfıra doğru doygunlaştırmaktadır. Bu ağın öğrenmeyi teşvik edebilmesini sağlamaktadır.

SELU, özellikle derin sinir ağlarında kullanıldığında bazı avantajlar sunabilmektedir. Ancak, bu fonksiyonun etkili olduğu durumlar belirli deney ve testlere dayanmaktadır. Bu nedenle kullanılacağı probleme bağlı olarak denemeler yapılması önerilmektedir. [14]

3.7 AdaDelta

AdaDelta, derin öğrenme modellerinin eğitiminde kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, öğrenme hızını manuel olarak ayarlamak yerine otomatik olarak uyarlanmış bir öğrenme hızı sağlamaktadır. AdaDelta, özellikle AdaGrad'ın dezavantajlarına çözüm oluşturmak üzere tasarlanmıştır.

AdaDelta algoritması, öğrenme hızını düzenleyerek bir tür RMSProp (Root Mean Square Propagation) algoritmasıdır.

AdaDelta'nın güncelleme kuralı aşağıdaki gibidir.

$$\text{accumulated\_grad} = \rho \times \text{accumulated\_grad} + (1 - \rho) \times \text{gradyan}^2$$
$$\text{delta} = -\frac{\sqrt{\text{accumulated\_param} + \epsilon}}{\sqrt{\text{accumulated\_grad} + \epsilon}} \times \text{gradyan}$$
$$\text{accumulated\_param} = \rho \times \text{accumulated\_param} + (1 - \rho) \times \text{delta}^2$$
$$\text{parametre} = \text{parametre} + \text{delta}$$

Bu formülde,

- accumulated_grad, gradyan karelerinin birleştirilmiş etkisini temsil etmektedir.
- accumulated_param, parametre güncellemelerinin karelerinin birleştirilmiş etkisini temsil etmektedir.
- ρ, birleştirme faktörüdür.
- ϵ, sıfıra bölme hatasını önlemek için küçük bir sabittir.
- gradyan, model parametrelerinin gradyanını temsil etmektedir.

AdaDelta, delta adlı bir değişken kullanarak öğrenme hızını otomatik olarak düzenlemektedir.

AdaDelta, öğrenme hızını düzenlemek için hiper parametre kullanımını azaltmaktadır. AdaDelta, geçmiş gradyan karelerini toplamak yerine bir birleştirme faktörü kullanarak bellek kullanımını azaltmaktadır.

AdaDelta, bazı diğer optimizasyon algoritmalarına göre daha hesaplama maliyetli olabilmektedir. Başlangıçta accumulated_grad ve accumulated_param değerlerinin belirlenmesi için özel bir hiper parametre gerektirmektedir.

AdaDelta, özellikle öğrenme hızını manuel olarak ayarlamak istemeyen kullanıcılar için uygun bir seçenek olabilmektedir. Ancak, özel başlangıç hiper parametreleri ve hesaplama maliyeti göz önüne alındığında dikkatlice seçilmelidir.[15]

3.8 AdaGrad

AdaGrad (Adaptive Gradient Descent), derin öğrenme modellerinin eğitiminde kullanılan bir optimizasyon algoritmasıdır. AdaGrad, her parametre için özelleştirilmiş bir öğrenme hızı sağlamak amacıyla her parametre için gradyanların geçmiş karelerinin birleştirilmiş etkisini kullanılmaktadır.

AdaGrad algoritması, her bir parametre için bir öğrenme hızı belirlemektedir. Bu öğrenme hızı, daha önceki gradyanların karelerinin toplamına bölünerek belirlenmektedir. Güncelleme kuralı aşağıdaki gibidir.

$$\text{accumulated\_grad} = \text{accumulated\_grad} + (\text{gradyan})^2$$
$$\text{parametre} = \text{parametre} - \frac{\text{learning rate}}{\sqrt{\text{accumulated\_grad}+\epsilon}} \times \text{gradyan}$$

Bu formülde,

- accumulated_grad, gradyan karelerinin birleştirilmiş etkisini temsil etmektedir.
- learning rate, öğrenme hızını temsil etmektedir.
- ϵ, sıfıra bölme hatasını önlemek için küçük bir sabittir.
- gradyan, model parametrelerinin gradyanını temsil etmektedir.

AdaGrad, her bir parametre için özelleştirilmiş bir öğrenme hızı sağlar. Bu, modelin farklı parametreleri için uygun hızda güncellenmesini sağlamaktadır. AdaGrad, öğrenme hızını doğru bir şekilde ayarlamak için düşük hiper parametre hassasiyetine sahiptir.

Bellek kullanımı bakımında AdaGrad, geçmiş gradyan karelerini topladığı için bellek kullanımı sorunu ortaya çıkabilmektedir. Bu da büyük modellerde bellek sorunlarına yol açabilmektedir. Eğitim sürecinin ilerlemesiyle birlikte accumulated_grad artmakta ve bu da öğrenme hızının zamanla düşmesine neden olabilmektedir. Bu durum, eğitim sürecinin sonlarına doğru öğrenme hızının çok küçük olmasına neden olabilmektedir.

AdaGrad, özellikle seyrek veri setleri ve doğrusal olmayan modellerle iyi çalışabilmektedir. Ancak, bellek kullanımı ve öğrenme hızının zamanla düşmesi gibi dezavantajları bulunmaktadır. Bu nedenle, farklı optimizasyon stratejileriyle karşılaştırarak model ve veri seti özelliklerine uygun bir seçim yapmak önemlidir.[16]

3.9 Adam (Adaptive Moment Estimation)

Adam, derin öğrenme modellerinde yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Bu algoritma hem momentum hem de RMSProp algoritmalarının avantajlarını birleştirerek adaptif bir öğrenme hızı sağlamaktadır.

Adam algoritması, iki hareketli ortalama kullanılmaktadır. Biri gradyanların birinci momentini (momentum), diğeri gradyanların ikinci momentini (RMSProp) kullanmaktadır. Bu hareketli ortalama değerleri ile her adımda güncellenmektedir. Bu değerlerle birleştirilerek her parametre için özelleştirilmiş bir öğrenme hızı oluşturulmaktadır.

Adam'ın güncelleme kuralları aşağıdaki gibidir.

$$m = \beta_1 \times m + (1 - \beta_1) \times \text{gradiyan}$$
$$v = \beta_2 \times v + (1 - \beta_2) \times \text{gradiyan}^2$$

Bu değerlerin düzeltilmiş hali de şu şekildedir.

$$\hat{m} = \frac{m}{1-\beta_1^t}$$
$$\hat{v} = \frac{v}{1-\beta_2^t}$$

t burada, o anki adım numarasını temsil etmektedir.

Son olarak, parametre güncellemesi şu şekilde yapılmaktadır.

$$\text{parametre} = \text{parametre} - \frac{\text{learning rate} \times \hat{m}}{\sqrt{\hat{v}+\epsilon}}$$

Bu formülde,

- β_1 ve β_2 momentum ve RMSProp'un hareketli ortalama faktörleridir.
- learning rate, öğrenme hızını temsil etmektedir.
- ϵ, sıfıra bölme hatasını önlemek için küçük bir sabittir.

Adam, her parametre için ayrı bir öğrenme hızı sağlar ve bu, eğitim sürecinde daha iyi bir performans sağlayabilmektedir. Hareketli ortalamalar, gradyanlardaki gürültüyü düzenler ve daha stabil bir eğitim süreci sağlamaktadır. Adam, diğer optimizasyon algoritmalarına göre daha az hiper parametre hassasiyetine sahiptir.

Özellikle büyük modellerde Adam, geçmiş gradyanların karesinin hareketli ortalamasını tuttuğu için bellek kullanımı bir dezavantaj olabilmektedir. Bazı durumlarda, eğitim sürecinin sonlarına doğru öğrenme hızının çok hızlı bir şekilde azalmasına neden olabilmektedir.

Adam, genellikle birçok uygulamada iyi performans gösteren etkili bir optimizasyon algoritmasıdır. Ancak, model ve veri seti özelliklerine bağlı olarak farklı optimizasyon stratejileri tercih edilebilmektedir.[17]

3.10 Momentum

Momentum, derin öğrenme modellerinin eğitiminde kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, gradyan inişini hızlandırmak ve daha hızlı yakınsak sağlamak amacıyla gradyan güncellemelerine bir momentum terimi eklemektedir.

Momentum algoritması, her adımda bir momentum terimini ekleyerek model parametrelerini güncellemektedir. Momentum terimi, önceki adımlardaki gradyanların birleştirilmiş etkisini temsil etmektedir.

Momentum ‘un güncelleme kuralı aşağıdaki gibi gösterilmektedir.

$$\text{velocity} = \beta \times \text{velocity} + \text{learning rate} \times \text{gradyan}$$
$$\text{parametre} = \text{parametre} - \text{velocity}$$

Bu formülde,

- β, momentum teriminin ağırlıklandırma faktörüdür.
- velocity, momentum terimini temsil etmektedir.
- learning rate, öğrenme hızını temsil etmektedir.
- gradyan, model parametrelerinin gradyanını temsil etmektedir.

Hızlandırılmış yakınsak ile Momentum, gradyan inişini hızlandırarak daha hızlı bir yakınsak elde etmeye yardımcı olabilmektedir. Azalmış dalgalanma ile Momentum, gradyanlardaki dalgalanmayı azaltarak daha düzenli bir eğitim süreci sağlamaktadır. Ayrıca yerel minimumlara karşı daha dirençliliği ile Momentum, yerel minimumlara karşı daha dirençli olabilmekte ve bu sayede modelin daha iyi genelleme yapmasına yardımcı olabilmektedir.[18]

Hiper parametre seçimi ile Momentumun başarısı, öğrenme hızı ve momentum terimi ağırlıklandırma faktörü gibi hiper parametre değerlerine bağlıdır. Bu değerlerin doğru seçilmesi önemli olabilmektedir. Ayrıca dengeleme zorluğu ile

Hiper parametrelerin dengeli bir şekilde seçilmesi zor olmaktadır. Örneğin, çok yüksek bir momentum değeri aşırı dalgalanmaya neden olabilmektedir.

Momentum, genellikle gradyan inişinin daha hızlı ve daha düzenli bir şekilde ilerlemesini sağlamak için kullanılan etkili bir optimizasyon algoritmasıdır. Ancak, uygun hiper parametrelerin seçilmesi ve dengeli bir şekilde ayarlanması önemlidir.[19]

3.11 Nadam

Nadam (Nesterov-accelerated Adaptive Moment Estimation), derin öğrenme modellerinin eğitiminde kullanılan bir optimizasyon algoritmasıdır. Nadam, hem momentum hem de AdaGrad'ın avantajlarını birleştirir ve Nesterov hızlandırılmış gradyan inişini kullanılmaktadır.

Nadam algoritması, hızlandırılmış gradyan inişinin bir varyasyonunu kullanılmaktadır. Bu varyasyon, Nesterov Momentum ve AdaGrad'ın bir kombinasyonunu içermektedir.

Nadam'ın güncelleme kuralı aşağıdaki gibidir.

$$\mathrm{m} = \beta_1 \times \mathrm{m} + (1 - \beta_1) \times \mathrm{gradyan}$$
$$\mathrm{v} = \beta_2 \times \mathrm{v} + (1 - \beta_2) \times \mathrm{gradyan}^2$$
$$\mathrm{parametre} = \mathrm{parametre} - \frac{\mathrm{learning\ rate}}{\sqrt{\mathrm{v}+\epsilon}} \times (\beta_1 \times \mathrm{m} + \frac{(1-\beta_1)\times \mathrm{gradyan}}{1-\beta_1^t})$$

Bu formülde,

- m, momentum terimini temsil etmektedir.
- v, AdaGrad benzeri terimi temsil etmektedir.
- β_1 ve β_2, momentum ve AdaGrad terimlerinin ağırlıklandırma faktörleridir.

- t, güncelleme adımını temsil etmektedir.
- ϵ, sıfıra bölme hatasını önlemek için küçük bir sabittir.
- gradyan, model parametrelerinin gradyanını temsil etmektedir.

Hızlandırılmış konvergens ile Nadam, momentum ve AdaGrad'ın avantajlarını birleştirerek daha hızlı bir yakınsak sağlayabilmektedir. Daha iyi genelleme ile Nesterov Momentum, genellikle daha iyi genelleme performansına katkıda bulunabilmektedir. Öğrenme hızı dengelemesi konusunda Nadam, öğrenme hızını otomatik olarak ayarlar, bu da hiper parametre seçimi konusunda daha az hassasiyet gerektirmektedir.

Nadam, hesaplama maliyeti daha yüksek olan bir algoritmadır. Hiper parametre hassasiyeti ile Nadam'ın başarısı, β_1, β_2 ve ϵ gibi hiper parametre değerlerine bağlıdır.

Nadam, genellikle karmaşık problemlerde ve büyük veri setlerinde iyi performans göstermektedir. Ancak, hesaplama maliyeti nedeniyle küçük veri setlerinde veya kaynak sınırlı ortamlarda tercih edilmeyebilmektedir.[9]

3.12 Sigmoid

Sigmoid aktivasyon fonksiyonu, genellikle binary sınıflandırma problemlerinde kullanılan bir tür aktivasyon fonksiyonudur. Sigmoid fonksiyonu, girdiyi 0 ile 1 arasında bir değere dönüştürür ve genellikle çıkış katmanındaki bir sinir hücresinde kullanılmaktadır. Bu fonksiyon, lojistik fonksiyon olarak da adlandırılmaktadır.

Sigmoid aktivasyon fonksiyonu matematiksel olarak aşağıdaki gibi gösterilmektedir.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Bu formülde,

- $\sigma(x)$, sigmoid fonksiyonunu temsil etmektedir.
- e, Euler sayısı (yaklaşık olarak 2.71828).
- x, girdi değeridir.

Sınırlı çıkış aralığı ile Sigmoid fonksiyonu, herhangi bir gerçel sayıyı (negatif veya pozitif) 0 ile 1 arasında bir değere dönüştürür. Bu özellik, özellikle binary sınıflandırma problemlerinde kullanışlıdır, çünkü çıkışı olasılık olarak yorumlamak mümkündür. Düz alan duyarsızlığı [20](Vanishing Gradient) ile birlikte Sigmoid fonksiyonunun türevi, girişin mutlak değerinin büyüklüğüne bağlı olarak çok küçük olabilmektedir. Bu durum ile geriye doğru yayılım sırasında gradiyanın kaybolmasına neden olabilmektedir. Bu da derin sinir ağlarında eğitim sırasında sorunlara yol açabilmektedir. Sigmoid fonksiyonunun çıkışı, girişin negatif veya pozitif olmasına bağlı olarak asimetrik olmaktadır. Bu da ağın öğrenmesini yavaşlatabilmektedir.

Sigmoid aktivasyon fonksiyonu, genellikle çıkış katmanındaki binary sınıflandırma problemlerinde kullanılmaktadır. Ancak, derin sinir ağlarında daha iyi performans elde etmek için başka aktivasyon fonksiyonları da tercih edilebilmektedir. Özellikle ReLU (Rectified Linear Unit) gibi fonksiyonlar tercih edilmektedir.

3.13 ELU (Exponential Linear Unit)

Derin öğrenme modellerinde kullanılan bir aktivasyon fonksiyonudur. ELU, negatif girişlere karşı sıfırdan farklı bir eğriye sahiptir. Bu sayede ReLU tabanlı aktivasyon fonksiyonlarının bazı sorunlarına çözüm sunmayı amaçlamaktadır.

ELU fonksiyonu matematiksel olarak aşağıdaki gibidir.

$$f(x) = \begin{cases} \alpha(exp(x)-1) & \text{for} \quad x \leq 0 \\ x & \text{for} \quad x > 0 \end{cases} f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x \leq 0 \\ 1 & \text{for} \quad x > 0 \end{cases}$$

$$\begin{array}{ll} x & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise } endcases \end{array}$$

Bu formülde,

- $(\mathrm{ELU}(x))$, ELU fonksiyonunu temsil etmektedir.
- (x), girdi değeridir.
- (α), küçük bir pozitif sabittir (genellikle 1).

Negatif girişlere duyarlılığı ile ELU, negatif girişler için sıfırdan farklı bir eğri kullanmaktadır. Bu sayede negatif girişlere karşı daha duyarlıdır ve "dying ReLU" problemiyle başa çıkmaktadır. ELU fonksiyonu, negatif girişlerde bile sürekli ve türevlenebilir bir eğriye sahiptir. Bu, modelin eğitilmesini ve geriye doğru yayılımı daha düzgün hale getirebilmektedir. Asimetrik davranış konusunda ELU, negatif girişlere karşı sıfırdan farklı bir eğri kullanarak asimetrik bir davranışa sahiptir olabilmektedir.

ELU, genellikle ReLU ve türevleri gibi diğer aktivasyon fonksiyonlarıyla karşılaştırılarak kullanılmaktadır. ELU'nun avantajları, negatif girişlere karşı daha toleranslı olması ve bazı eğitim sorunlarına çözüm sunmasıdır. Ancak, her aktivasyon fonksiyonu gibi, kullanılacak veri seti ve probleme bağlı olarak performansı değişebilmektedir.[21]

3.14 ReLU (Rectified Linear Unit)

ReLU (Rectified Linear Unit), derin öğrenme modellerinde sıkça kullanılan bir aktivasyon fonksiyonudur. ReLU fonksiyonu, giriş değeri pozitifse, giriş değerini

doğrudan çıkış olarak alabilmektedir. Eğer giriş değeri negatifse, çıkış değerini sıfır yapmaktadır.[22]

ReLU fonksiyonu matematiksel olarak aşağıdaki gibi gösterilmektedir.

$$\mathrm{ReLU}(x) = \max(0, x)$$

Bu formülde:

- $\mathrm{ReLU}(x)$, ReLU fonksiyonunu temsil etmektedir.
- x, girdi değeridir.

Basit ve etkili olan ReLU, hesaplama açısından basit ve hızlı olmaktadır. Bu, özellikle büyük veri setleri ve karmaşık modellerle çalışırken avantajlıdır. Sparsity teşviki ile ReLU, negatif girişleri sıfıra düşürdüğü için, ağın aktivasyon haritasını seyrek hale getirebilmektedir. Bu da daha az bellek kullanımına ve daha etkili öğrenmeye katkıda bulunabilmektedir. Vanishing Gradient sorununu azaltma hakkında Sigmoid ve tanh gibi aktivasyon fonksiyonları, geriye doğru yayılım sırasında gradiyan kaybına neden olabilen küçük türevlere sahiptirler. ReLU'nun türevi, pozitif girişler için her zaman 1 olduğu için bu sorunu azaltabilmektedir.

Ancak bununla birlikte ReLU'nun bazı dezavantajları da bulunmaktadır.[23]

Dying ReLU problemi gibi ReLU'nun negatif girişlere sıfır çıkış vermesi, bazı nöronların eğitim sırasında hiç aktive olmamasına ve "ölmesine" neden olabilmektedir. Bu durumu çözmek için değişiklikler yapılan türevleri olan Leaky ReLU gibi varyasyonlar kullanılabilmektedir. Exploding Gradient Sorunu ile ReLU, girişin pozitif olması durumunda sınırlı bir çıkış aralığına sahip olmadığı için, büyük girişlerle çalışırken "exploding gradient" sorununa neden olabilmektedir. ReLU, negatif girişlere karşı tamamen kapalıdır, bu da asimetrik davranışa neden olabilmektedir.[24]

Günümüzde, ReLU ve türetilmiş varyasyonları (Leaky ReLU, Parametric ReLU, Exponential Linear Unit - ELU vb.) birçok derin öğrenme modelinde yaygın olarak kullanılmaktadır. [25]

3.15 Stokastik Gradyan İnişi (SGD)

Stokastik Gradyan İnişi (SGD), derin öğrenme modellerinin eğitiminde yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, model parametrelerini güncellerken eğitim verilerinden rastgele örnekler kullanılmaktadır.

SGD, her bir eğitim adımında bir mini-batch (küçük bir örnek alt kümesi) seçer ve sadece bu mini-batch üzerinde gradyan hesaplar. Bu gradyan kullanılarak model parametreleri güncellenmektedir. Bu süreç, tüm eğitim verisi üzerinde tamamlanana kadar tekrarlanmaktadır.

SGD'nin güncelleme kuralı aşağıdaki gibi gösterilmektedir.

parametre=parametre−learning rate×gradyan

Bu formülde,

- learning rate, öğrenme hızını temsil etmektedir.
- gradyan, model parametrelerinin gradyanını temsil etmektedir.

Hafıza Verimliliği ile SGD, her adımda sadece bir mini-batch kullanarak hafıza verimliliği sağlamaktadır. Bu özellik, büyük veri setleri üzerinde çalışırken önemli olmaktadır. SGD, her adımda yeni veri örnekleri kullanarak online öğrenme için uygundur. Çünkü her adımda sadece bir mini-batch kullanılmaktadır. Gradyan hesaplama ve model güncelleme işlemleri daha hızlı gerçekleştirmektedir.[20]

SGD, gradyanların her mini-batch'te değişkenlik göstermesi nedeniyle dalgalanmaya neden olabilmektedir. Bu, eğitim sürecini daha gürültülü hale getirebilmektedir. Konverjans Hızı ile SGD, düzensiz gradyanlar ve gürültülü güncellemeler nedeniyle diğer optimizasyon algoritmalarına kıyasla daha yavaş bir yakınsak hızına sahip olabilmektedir. Hiper parametre hassasiyeti ile de öğrenme hızı gibi hiper

parametrelerin seçimi, SGD'nin başarısını etkiler ve bu seçim problemine hassas olabilmektedir.

SGD, hafıza verimliliği ve hızlı güncelleme avantajları nedeniyle geniş bir uygulama alanına sahiptir. Ancak, dalgalanma ve konverjans hızı gibi dezavantajları bulunmaktadır. Bu nedenle bu dezavantajlar göz önüne alınarak model ve veri seti özelliklerine uygun bir optimizasyon stratejisi seçilmelidir.[26]

4. BÖLÜM

4.1 Savaş Uçaklarında Görüntü Tanımanın Önemi

Savaş uçaklarında model, tip, tasarımsal değişiklikler beraberinde kabiliyetlerinin yükselmesi ile Türkiye de dahil bazı güçlü ülkelerin kendi ürettikleri taktiksel muhabere uçaklarında küresel bir artış bulunmaktadır. Uçakların tespit edilerek akıllıca seçme bilinci, doğru bir yönlendirme için temel faktörlerden birisidir. Savaş uçakları için görüntü tanıma sistemi tasarlamak için, birçok uçak görüntüsü bir mobil cihazdan yakalanmaktadır.

4.2 Askeri Hedeflerin Önemi ve Savaş Alanı Nesnelerinin Tespiti

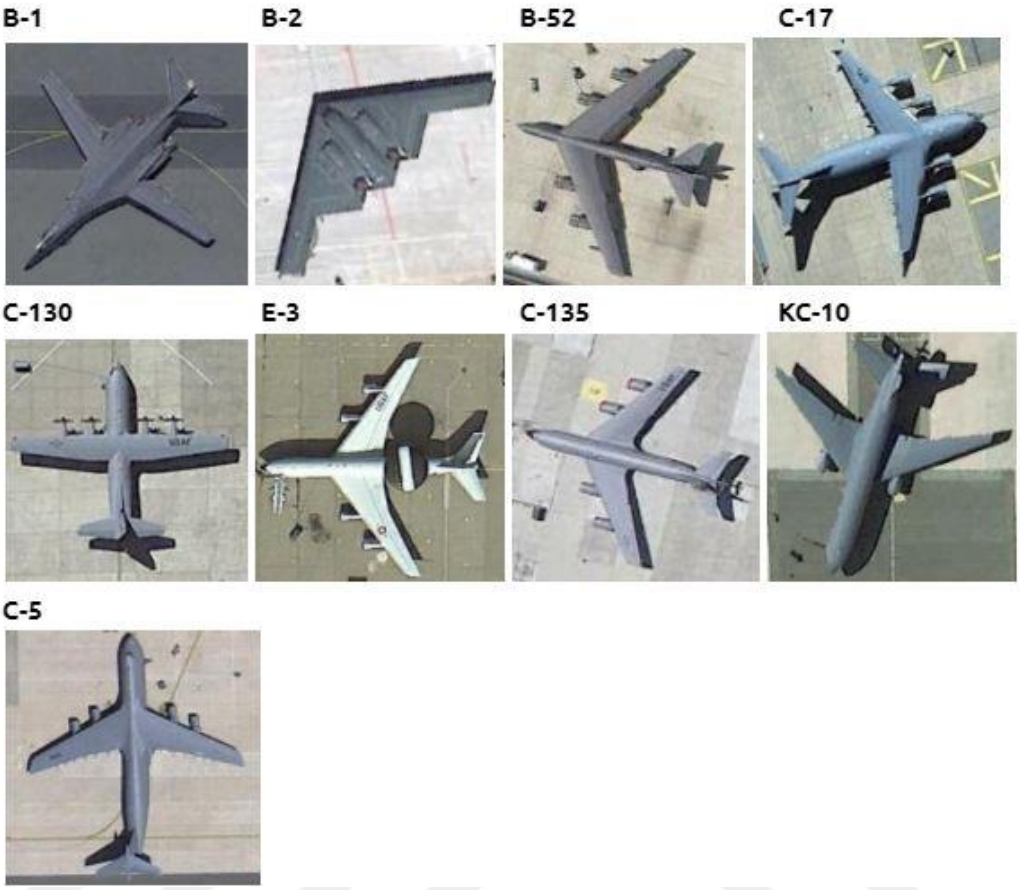
Bugün, dünyanın dört bir yanındaki birçok askeri şube, İstihbarat, Gözetleme ve Keşif için Bilgisayarla Görme 'ye (ISR) yatırım yapıyor. Bu alanla ilgili bazı uygulamalar arasında sosyal medyada askeri araçların tanınması[1], kamufle edilmiş askeri hedeflerin tanınması [2] ve savaş alanı nesnelerinin tespiti [3] yer almaktadır. Amerika Birleşik Devletleri gibi askeri süper güçler, yapay zekâ ve derin öğrenmeye büyük yatırımlar yapıyor ve görüntü tanıma ana uygulamalardan biri olarak görülüyor [4]. Bu alanın temel amacı, ISR yeteneklerini geliştirmektir. Günümüzde, savaş ve keşif dronları, dünyanın dört bir yanındaki modern ordularda giderek daha önemli ve önemli hale geliyor. İHA ve SİHA 'larda en önemli unsurlardan biri de yüksek irtifalardan görüntüleme kabiliyetleridir. Bununla birlikte, bugün olduğu gibi, keşif ve uzaktan istihbarat toplama hala uydu görüntüleri aracılığıyla yapılmaktadır. Bu proje, Görüntü İşleme'nin ordular tarafından ISR amaçları için nasıl kullanılabileceğinin temellerine odaklanmaktadır.

4.3 Veri Setleri ve Kategorileri

Veri (veriseti) 9 farklı askeri uçaktan oluşan bir veri setinin yanı sıra, hava üslerinin beton asfaltı veya dünyadaki belirli kemik sahaları ve hava üsleri için kumlu zeminler gibi çeşitli arka planlar için bir veri seti geliştirdik. Genel olarak, veri seti, her biri 120 x 120 olan 6.300 görüntüden oluşur ve her uçak tipi ~ 610 - 660 görüntü içermektedir.

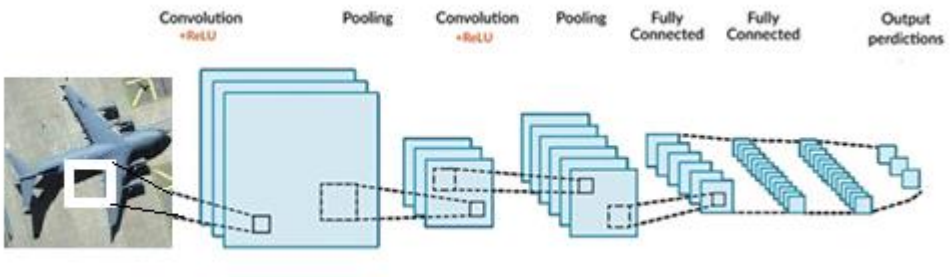
Tablo 4.1: Veri boyutu, hangi veri, kaç tip var, sınıf kategorisi

Görüntü Tipi	Resim Sayısı
B-1	620
B-2	663
B-52	607
C-5	616
C-17	667
C-130	632
C-135	616
E-5	626
KC-10	638
BareLand (Yeryüzü)	615



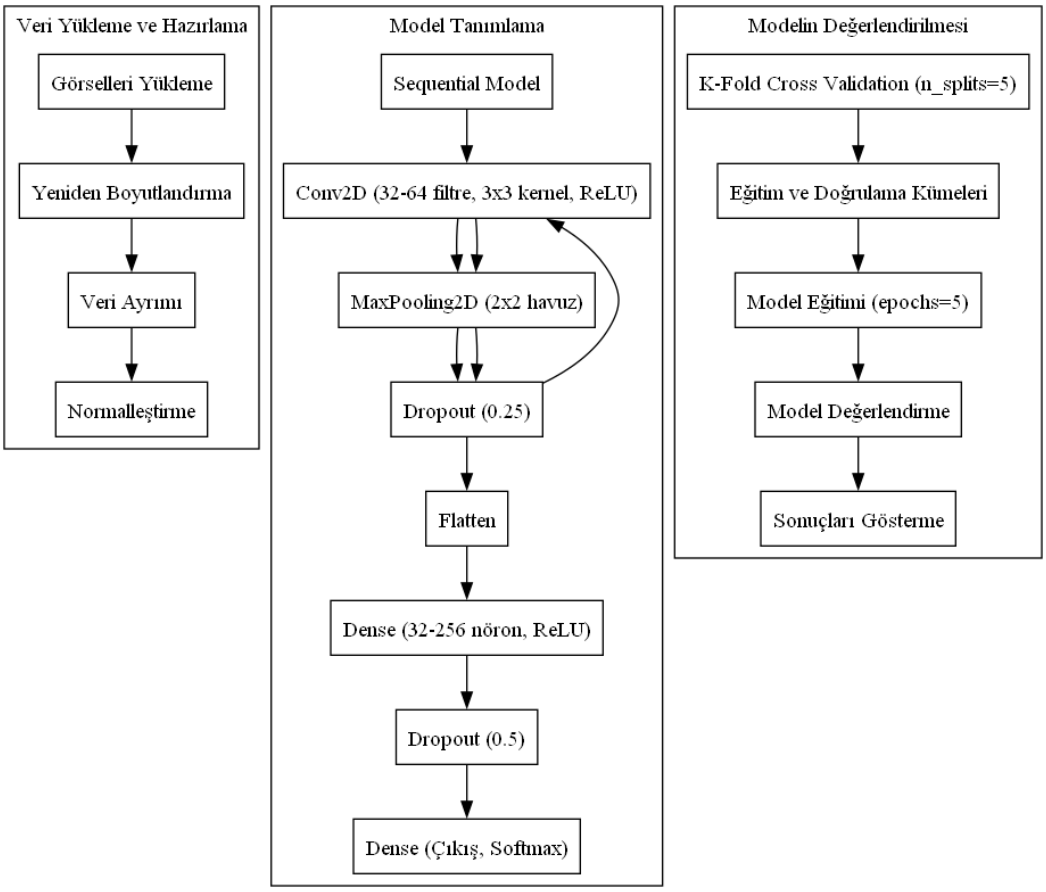
Şekil 4.1: Uçak Çeşitleri ve Örnek Görselleri

Model, biri nispeten küçük veri kümemiz, ikincisi ise çapraz doğrulama kaybını mümkün olduğunca en aza indirmek olmak üzere iki önemli hususa dayalı olarak tasarlanmıştır. Aşırı uyumu önlemek için, küçük bir veri kümesi boyutu göz önüne alındığında daha az karmaşıklığa sahip nispeten küçük bir model çok önemlidir. Evrişimli sinir ağımız aşağıdakilerden oluşur.

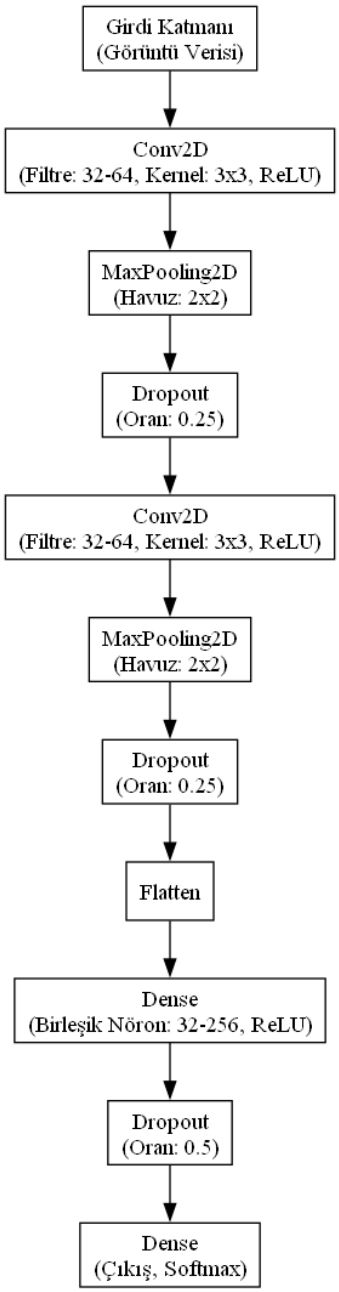


Şekil 4.2: Tipik CNN Model Mimarisi

4.4 Görüntü İşlemede Uygulanan Yeni Algoritma ve Modeller



Şekil 4.3: Görüntü işlemede kullanılan model örneği



Şekil 4.4: Görüntü işleme dizininde temel olarak kullanılan adımlar

Önceki Model katmanlarına Tablo 4.2 de bakacak olursak;

Tablo 4.2: Model Çıktısı: Model: "sequential_8"

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 128, 128, 32)	896
activation_40 (Activation)	(None, 128, 128, 32)	0
max_pooling2d_24 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_25 (Conv2D)	(None, 62, 62, 64)	18496
activation_41 (Activation)	(None, 62, 62, 64)	0
max_pooling2d_25 (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_26 (Conv2D)	(None, 29, 29, 32)	18464
activation_42 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_26 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_8 (Flatten)	(None, 6272)	0
dense_16 (Dense)	(None, 450)	2822850
activation_43 (Activation)	(None, 450)	0
dropout_8 (Dropout)	(None, 450)	0
dense_17 (Dense)	(None, 10)	4510
activation_44 (Activation)	(None, 10)	0

Total params: 2865216 (10.93 MB)

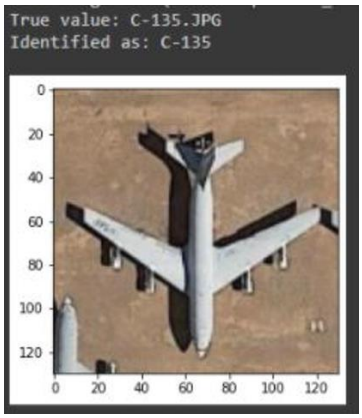
Trainable params: 2865216 (10.93 MB)

Non-trainable params: 0 (0.00 Byte)

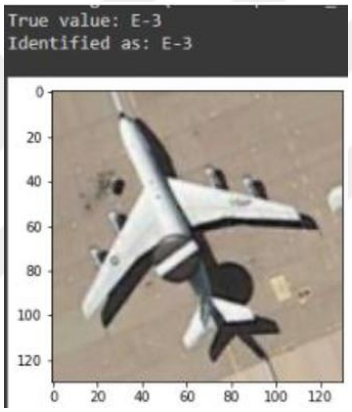
Keras tarafından sağlanan hiper parametre ayarlama fonksiyonunun yardımıyla, Şekil 4.15'te en yüksek çapraz doğrulama doğruluğunu ve en düşük çapraz doğrulama kayıp değerini elde etmeye odaklanan bir model bulunmuştur. Modelin özeti Şekil 4.5'te görülebilmektedir. Modelin elde edebildiği en yüksek çapraz doğrulama doğruluğu ~ %96'dir ve elde edilen en düşük çapraz doğrulama doğruluğu ~ 0.3322'dir. Eğitim doğruluğu ~ %98 ve eğitim kaybı ~ 0.0610'dur. Model, seyrek kategorik çapraz entropi kaybı yöntemi kullanılarak 0.0001 öğrenme oranına sahip Adam optimizer ile derlenmiştir. Model, 30 dönem, 64 parti boyutu ve .30 doğrulama bölümü ile yapılmıştır.

4.5 Modelin Pratik Yetenekleri

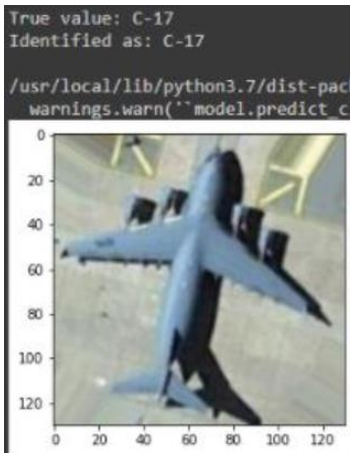
Modelin pratik yeteneklerini, dünya çapındaki hava kuvvetleri üslerinin çeşitli geniş gövdeli anlık görüntüleri üzerinde tahmin işlevini kullanarak test ettik. Bu, modelin iki ana yeteneğini test etmek için yapıldı; modelin her türlü uçağı başarılı bir şekilde tanımlama ve tespit etme yeteneği ve tespit edilen uçağın ait olduğu kategoriyi belirleme yeteneği. Modelin uçağı tespit etme yeteneği Şekil 4.5, 4.6, 4.7'de görülmektedir. Ayrıca modelin bir uçağı doğru bir şekilde sınıflandırma yeteneği Şekil 4.8, 4.9, 4.10'de görülmektedir. Test aşamamız sırasında şunu fark ettik: model belirli uçak türlerini yanlış sınıflandırma eğilimindeydi. Daha derinlemesine baktığımızda, modelin yüksek derecede benzerliğe sahip uçak türleri arasında ayrım yapmakta genellikle zorlandığını fark ediyoruz. Örneğin, E-3 ve C-135'in her ikisi de aşağı yukarı aynı ölçülere ve özelliklere sahip, son derece benzer uçaklardır. Bu, modelin, diğer uçaklar arasında yaygın olabilecek uçağın genel şeklinden ziyade benzersiz özelliklerini daha fazla tanımlamak için eğitilmesi gerektiği anlamına gelir. Bu, veri kümemizi genişlettiğimiz ve pikselleştirme, veri artırma ve dengeleme açısından kalitesini iyileştirdiğimiz takdirde daha karmaşık bir model aracılığıyla başarılabilir.



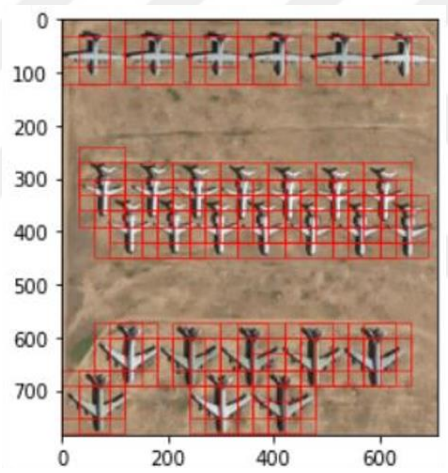
Şekil 4.5: C-135 Uçağının Tespit Edilmesi



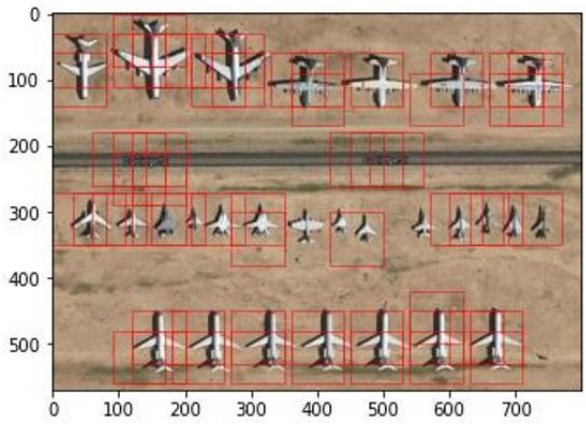
Şekil 4.6: E-3 Uçağının Tespit Edilmesi



Şekil 4.7: C-17 Uçağının Tespit Edilmesi



Şekil 4.8: Uçakların Yeryüzünde Tespit Edilebilmesi



Şekil 4.9: Uçakların Yeryüzünde Hatalarla Tespit Edilebilmesi

Şekil 4.10: Uçakların Yeryüzünde Tespit Edilebilmesi Örneği

4.6 Modelin Önceki Modeller ile Karşılaştırması

Bir önceki Modelden değiştirilmiş yerler belirtilerek katmanlarına Tablo 4.3'e bakacak olursak;

Tablo 4.3: Yeni Model Çıktısı: Model: "sequential_9"

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 128, 128, 32)	896
activation_40 (Activation)	(None, 128, 128, 32)	0
max_pooling2d_24 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_25 (Conv2D)	**(None, 128, 128, 64)**	**1,792**
activation_41 (Activation)	(None, 62, 62, 64)	0
max_pooling2d_25 (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_26 (Conv2D)	**(None, 62, 62, 64)**	**36,928**
activation_42 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_26 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_8 (Flatten)	**(None, 61504)**	**0**
dense_16 (Dense)	**(None, 128)**	**7,872,640**
activation_43 (Activation)	(None, 450)	0
dropout_8 (Dropout)	(None, 450)	0
dense_17 (Dense)	**(None, 10)**	**1,290**
activation_44 (Activation)	(None, 10)	0

Total params: 23,737,952 (90.55 MB)

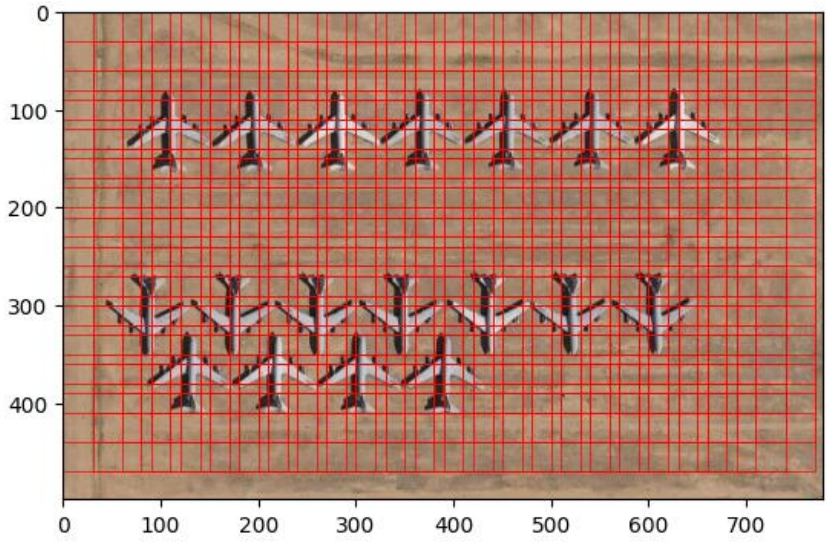
Trainable params: 7,912,650 (30.18 MB)

Optimizer params: 15,825,302 (60.37 MB)

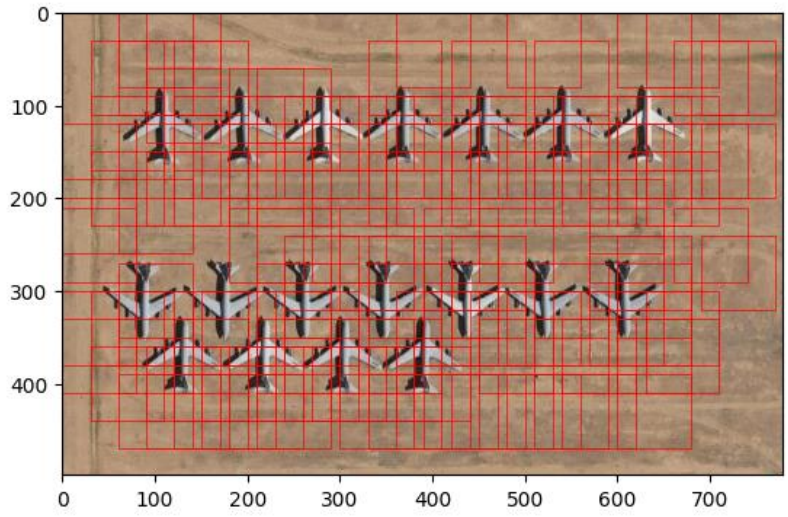
4.7 Test Modellerinde Savaş Uçaklarının Yeryüzünde Tespitinin Yapılışı

Test yapılan modellerde uçakların yeryüzünde tespit edilememesi ile ilgili alınan hatalar ve hataların azaltılarak uçakların tespit edilmesine kadar oluşturulan modeller Şekil 4.11,4.12,4.13 ve 4.14'te görülmektedir.

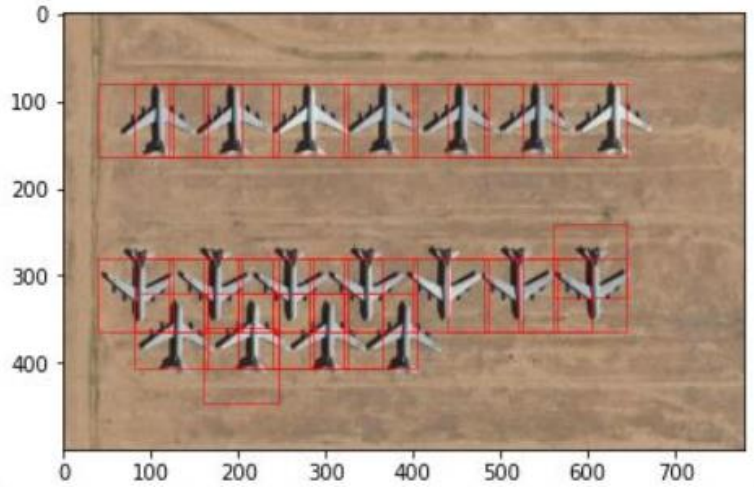
Şekil 4.11: Bir Modelde Uçakların Yeryüzünde Tespit Edilememesi



Şekil 4.12: Bir Modelde Uçakların Yeryüzünde Tespit Edilememesi ve Hataları



Şekil 4.13: Bir Modelde Uçakların Yeryüzünde Tespit Edilememesi ve Hata Azalması



Şekil 4.14: Bir Modelde Uçakların Yeryüzünde Tespit Edilmesi

Karışıklık Matriksi tahmin edilen değerler ile gerçek değerlerle kıyası yapılabilmektedir. Matris içinde verilenler ise, 1: TRUE ve 0: FALSE olacak şekilde yapılmaktadır.[27]

TP (True Positive): 1 olarak sınıflandırdığımız ve gerçekten de 1 olan değerlerin sayısıdır.

FP (False Positive): 1 olarak sınıflandırdığımız fakat gerçekte 0 olan değerlerin sayısıdır.

TN (True Negative): 0 olarak sınıflandırdığımız ve gerçekten de 0 olan değerlerin sayısıdır.

FN (False Negative): 0 olarak sınıflandırdığımız fakat gerçekte 1 olan değerlerin sayısıdır.

Bu ölçümler Precision, Recall, F-Score, Sensitivity ve Specificity formülleriyle sağlanımaktadır.

Precision: Doğru sınıflandırılan verilerin oranını vermektedir.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Sadece pozitif değerlerden doğru sınıflandırılanların oranını vermektedir.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F-Score (F-measure): Precision ve Recall değerlerinin harmonik ortalamasını vermektedir.

$$\text{F-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Sensitivity: Recall ile aynıdır. Pozitif olarak sınıflandırılan verilerin gerçekteki pozitif verilere oranını vermektedir.

$$\text{Sensitivity} = \frac{TP}{TP + FP}$$

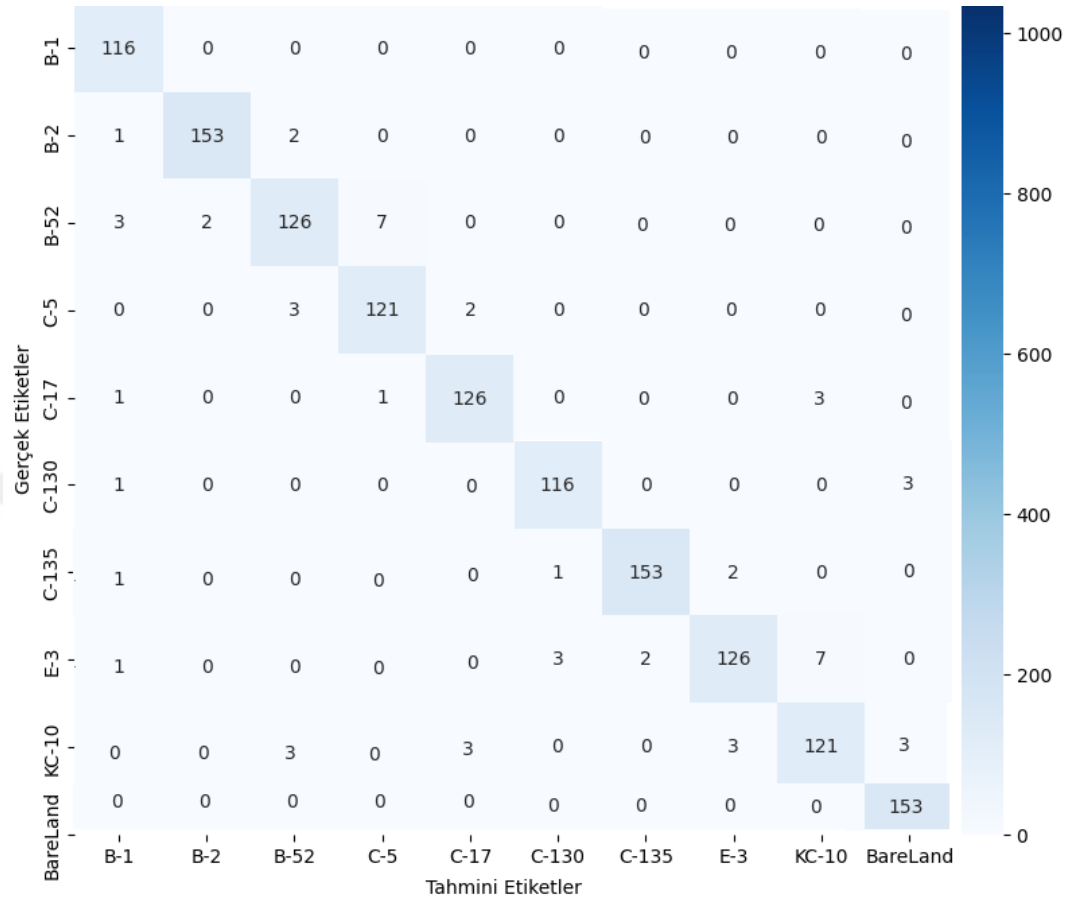
Specificity: Sadece negatif olarak sınıflandırılan verilerin gerçekteki negatif verilere oranını vermektedir.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Accuracy değeri ise doğruluk değerinin kendisidir ve aşağıdaki gibi hesaplanır.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Şekil 4.15: Karışıklık Matrisi



Tablo 4.4: Model Sonuçları

Model	Accuracy(%)	Recall	F1-Score	Precision	Testing Time
Model V3 (S.Khos,2021)	50.248756	0.502488	0.336569	0.253281	195.132124
Model V4 (S.Khos,2021)	51.641791	0.516418	0.351999	0.267142	200.798597
Model V5 (S.Khos,2021)	78.589232	0.785892	0.745684	0.753846	185.588668
Model V6 (S.Khos,2021)	95.637161	0.956372	0.953594	0.951565	190.526758
Model V7 (Unal,Z.,2024)	*98.808926*	*0.988089*	*0.985645*	*0.983299*	*140.357*
Model V8 (Unal,Z.,2024)	**99.125968**	**0.99126**	**0.990856**	**0.990536**	**438.777848**

Öncelikle, Conv2D katmanlarındaki filtre sayılarını ve kernel boyutları ayarlanmıştır. Ek olarak, Dense katmanındaki nöron sayısını optimize edebilerek ve eğitim sürecini daha iyi hale getirmek için erken durdurma (early stopping) ve öğrenme oranı azaltma (learning rate reduction) gibi callback'ler eklenebilmektedir.

4.8 Yapılan Değişiklikler ve Performans Değerlendirmesi

Tablo 4.4 de görüldüğü gibi Model v7 ve Model v8 de daha yüksek filtre sayıları ve her bir Conv2D katmanında padding='same' kullanılarak modelin kapasitesini arttırılmıştır. Son Dense katmanındaki nöron sayısını 450'den 512'ye çıkartılarak daha fazla öğrenme kapasitesi sağlanmıştır. Dropout oranı 0.65'ten 0.5'e düşürülmüştür. Model v7 her ne kadar doğruluğu düşük olduğu tespit edilse de test zamanının kısa olmasından dolayı en iyi performans olduğu değerlendirilmiştir. Bu performansları yakalayabilmek için Şekil 4.16, 4.17 ve 4.18'deki gibi bazı denemeler yapılmıştır.

Şekil 4.15 Karışıklık Matrisindeki oranlara bakıldığında Early Stopping ve ReduceLROnPlateau gibi callback'ler eğitim sırasında modelin aşırı öğrenmesini engelleyerek ve öğrenme oranını gerektiğinde azaltarak bu eğitim sürecini daha verimli hale getirmektedir. Epochs ve Patience Değerleri ile Epochs sayısını artırılarak ve patience değerleri belirlenmiştir. Bu değişikliklerle modelimiz daha iyi performans göstermiştir.

```
Trial 3 Complete [00h 04m 24s]
val_accuracy: 0.9494949579238892

Best val_accuracy So Far: 0.9494949579238892
Total elapsed time: 00h 12m 18s
7/7 [==============================] - 1s 176ms/step
7/7 [==============================] - 2s 284ms/step
7/7 [==============================] - 1s 184ms/step
7/7 [==============================] - 2s 319ms/step
7/7 [==============================] - 2s 244ms/step
  Model  5-CV Accuracy (%)    Recall  F1-Score  Testing Time (s)
0   CNN          99.207921  0.992079   0.99192        140.164372
```

Şekil 4.16: 1000 adet örnek ile test yapılması

```
Default search space size: 4
conv_1_filter (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 64, 'step': 16, 'sampling': 'linear'}
conv_2_filter (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 64, 'step': 16, 'sampling': 'linear'}
dense_1_units (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step': 32, 'sampling': 'linear'}
learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001], 'ordered': True}
13/13 [==============================] - 3s 199ms/step
13/13 [==============================] - 3s 197ms/step
13/13 [==============================] - 3s 202ms/step
13/13 [==============================] - 3s 193ms/step
13/13 [==============================] - 3s 203ms/step
  Model  5-CV Accuracy (%)    Recall  F1-Score  Testing Time (s)
0   CNN    96.608479  0.966085  0.966193        277.631849
```

Şekil 4.17: 2000 adet örnek ile test yapılması

```
Reloading Tuner from 1721763928/untitled_project/tuner0.json
Search space summary
Default search space size: 4
conv_1_filter (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 64, 'step': 16, 'sampling': 'linear'}
conv_2_filter (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 64, 'step': 16, 'sampling': 'linear'}
dense_1_units (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step': 32, 'sampling': 'linear'}
learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001], 'ordered': True}
19/19 [==============================] - 5s 253ms/step
19/19 [==============================] - 5s 247ms/step
19/19 [==============================] - 6s 300ms/step
19/19 [==============================] - 7s 368ms/step
19/19 [==============================] - 6s 338ms/step
  Model  5-CV Accuracy (%)   Recall  F1-Score  Testing Time (s)
0   CNN        93.577038  0.93577  0.932158        498.269661
```

Şekil 4.18: 3000 adet örnek ile test yapılması

SONUÇ VE GENEL DEĞERLENDİRME

Askeri kuvvetlere ait üslerin farklı geniş gövdeli ve kanatları bulunan uçaklarının görüntülerini kullanarak modelin tahmin yetenekleri test edilmiştir.

Modelin gerçek dünya koşullarında nasıl performans gösterdiğini görmek için, hava modelin iki ana yeteneğini değerlendirmeyi amaçlanmıştır. Farklı türdeki uçakları doğru bir şekilde tanımlayıp tespit etme yeteneğine bakılmıştır. Tespit edilen uçağın hangi kategoriye ait olduğunu belirleme yeteneğine bakılmıştır. Testler sırasında, modelin bazı uçak türlerinin benzer kanat ve gövde oranlarından dolayı yanlış sınıflandırma eğiliminde olduğunu fark edilmiştir. Daha detaylı bir inceleme sonucunda, modelin birbirine çok benzeyen uçak türlerini ayırt etmekte zorlandığını belirlenmiştir. Uçakların birbirine çok benzer ölçülere ve özelliklere sahiptir, bu yüzden model bu tür uçakları ayırt etmekte zorlanabilmektedir.

Bu durumda, modelin sadece genel uçak şekillerine üzerinden olmadığı, aynı zamanda uçakların benzersiz özelliklerine de daha fazla odaklanacak şekilde eğitilmesi gerçekleştirilmiştir.

Doğruluğu %99 küsür oranlara çıkartılmış olan sonuç ile daha iyi bir performansa ulaşmak için daha fazla veri kümesini genişleterek ve veri pikselleştirme, artırma ve dengeleme gibi tekniklerle verilerin kalitesini artırmak gerekmektedir. Bu sayede daha zor aşamalı bir modelle bu sorun çözülebileceği değerlendirilmektedir.

REFERANSLAR

[1] U. C. Aytaç, A. Güneş, and N. Ajlouni, "A novel adaptive momentum method for medical image classification using convolutional neural network," BMC Med Imaging, vol. 22, no. 1, 2022, doi: 10.1186/s12880-022-00755-z.

[2] A. Manna, N. Upasani, S. Jadhav, R. Mane, R. Chaudhari, and V. Chatre, "Bird Image Classification using Convolutional Neural Network Transfer Learning Architectures," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 3, 2023, doi: 10.14569/IJACSA.2023.0140397.

[3] A. Mohanty, G. Goldsztein, and R. Pellegrin, "Fish Species Image Classification Using Convolutional Neural Networks," *Journal of Student Research*, vol. 11, no. 3, 2022, doi: 10.47611/jsrhs.v11i3.3058.

[4] S. Phiphitphatphaisit and O. Surinta, "Deep feature extraction technique based on conv1d and lstm network for food image recognition," *Engineering and Applied Science Research*, vol. 48, no. 5, 2021, doi: 10.14456/easr.2021.60.

[5] Ö. D. Kılıç, M. E. Aydemir, and P. Öztürk Özdemir, "Uçak Görüntülerinin Sınıflandırılmasında Farklı Yapay Zekâ Algoritmalarının Performansı," 2019. doi: 10.36287/setsci.5.2.018.

[6] Y. Wang, Y. Chen, and R. Liu, "Aircraft Image Recognition Network Based on Hybrid Attention Mechanism," *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/4189500.

[7] T. Bakirman and E. Sertel, "A benchmark dataset for deep learning-based airplane detection: HRPlanes," *International Journal of Engineering and Geosciences*, vol. 8, no. 3, 2023, doi: 10.26833/ijeg.1107890.

[8] X. Yuan, D. Fu, and S. Han, “Vision-based aircraft pose estimation with dual attention module for global feature extraction in complex airport scenes,” *Visual Computer*, 2023, doi: 10.1007/s00371-023-03110-7.

[9] P. N. Tan, M. Steinbach, V. Kumar, and et al., *Introduction to Data Mining, First Edition*, no. September. 2014.

[10] F. Kong, C. Zhao, and S. Li, “Best-of-three-worlds Analysis for Linear Bandits with Follow-the-regularized-leader Algorithm,” in *Proceedings of Machine Learning Research*, 2023.

[11] A. Al Smadi, A. Mehmood, A. Abugabah, E. Almekhlafi, and A. M. Al-Smadi, “Deep convolutional neural network-based system for fish classification,” *International Journal of Electrical and Computer Engineering*, vol. 12, no. 2, 2022, doi: 10.11591/ijece.v12i2.pp2026-2039.

[12] Purwono, A. Ma’arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. U. Haq, “Understanding of Convolutional Neural Network (CNN): A Review,” *International Journal of Robotics and Control Systems*, vol. 2, no. 4, 2022, doi: 10.31763/ijrcs.v2i4.888.

[13] K. Lakhdari and N. Saeed, “A new vision of a simple 1D Convolutional Neural Networks (1D-CNN) with Leaky-ReLU function for ECG abnormalities classification,” *Intell Based Med*, vol. 6, 2022, doi: 10.1016/j.ibmed.2022.100080.

[14] E. Bonmati, “Automatic segmentation method of pelvic floor levator hiatus in ultrasound using a self-normalizing neural network,” *Journal of Medical Imaging*, vol. 5, no. 02, 2018, doi: 10.1117/1.jmi.5.2.021206.

[15] A. T. Azar *et al.*, “Automated System for Colon Cancer Detection and Segmentation Based on Deep Learning Techniques,” *International Journal of Sociotechnology and Knowledge Development*, vol. 15, no. 1, 2023, doi: 10.4018/IJSKD.326629.

[16] M. J. Uddin, Y. Li, M. A. Sattar, Z. M. Nasrin, and C. Lu, “Effects of Learning Rates and Optimization Algorithms on Forecasting Accuracy of Hourly Typhoon Rainfall: Experiments With Convolutional Neural Network,” *Earth and Space Science*, vol. 9, no. 3, 2022, doi: 10.1029/2021EA002168.

[17] E. M. Dogo, O. J. Afolabi, and B. Twala, “On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification,” *Applied Sciences (Switzerland)*, vol. 12, no. 23, 2022, doi: 10.3390/app122311976.

[18] E. Kiyak and G. Unal, “Small aircraft detection using deep learning,” *Aircraft Engineering and Aerospace Technology*, vol. 93, no. 4, 2021, doi: 10.1108/AEAT-11-2020-0259.

[19] Q. Gazawy, S. Buyrukoğlu, and Y. Yılmaz, “Convolutional neural network for pothole detection in different road and weather conditions,” *Journal of Computer & Electrical and Electronics Engineering Sciences*, vol. 1, no. 1, 2023, doi: 10.51271/jceees-0001.

[20] О. Ю. Гаваза, И. В. Балабанов, and Т. В. Балабанова, “Methods of calculating the aircraft wing characteristics with influence aeroelasticity,” *MECHANICS OF GYROSCOPIC SYSTEMS*, vol. 0, no. 29, 2015, doi: 10.20535/0203-377129201563837.

[21] N. Phukan, S. Mohine, A. Mondal, M. S. Manikandan, and R. B. Pachori, “Convolutional Neural Network-Based Human Activity Recognition for Edge Fitness and Context-Aware Health Monitoring Devices,” *IEEE Sens J*, vol. 22, no. 22, 2022, doi: 10.1109/JSEN.2022.3206916.

[22] L. Du, G. Li, H. Chang, and H. Hao, “Military applications of artificial intelligence,” in *Lecture Notes in Electrical Engineering*, 2020. doi: 10.1007/978-981-15-6978-4_122.

[23] T. Hiippala, “Recognizing military vehicles in social media images using deep learning,” in *2017 IEEE International Conference on Intelligence and*

Security Informatics: Security and Big Data, ISI 2017, 2017. doi: 10.1109/ISI.2017.8004875.

[24] C. Sun *et al.*, "A multi-model architecture based on deep learning for aircraft load prediction," *Communications Engineering*, vol. 2, no. 1, 2023, doi: 10.1038/s44172-023-00100-4.

[25] S. R. Dubey and S. Chakraborty, "Average biased ReLU based CNN descriptor for improved face retrieval," *Multimed Tools Appl*, vol. 80, no. 15, 2021, doi: 10.1007/s11042-020-10269-x.

[26] S. Suhirman, R. Rianto, I. Santosa, and R. Yunanto, "THE ENSEMBLE METHOD AND SCHEDULED LEARNING RATE TO IMPROVE ACCURACY IN CNN USING SGD OPTIMIZER," *Journal of Engineering Science and Technology*, vol. 18, no. 6, 2023.

[27] S. Dhulipala, F. F. Adedoyin, and A. Bruno, "Sign and Human Action Detection Using Deep Learning," *J Imaging*, vol. 8, no. 7, 2022, doi: 10.3390/jimaging8070192.