KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY SİNİR AĞI TABANLI GÖRÜNTÜ SINIFLANDIRMA TEKNİĞİ İLE X-RAY TARAMA GÖRÜNTÜLERİNDEN USB BELLEK İÇERENLERİN SINIFLANDIRILMASI

YÜKSEK LİSANS TEZİ

Ali HACIHAMZAOĞLU

EYLÜL 2023
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY SİNİR AĞI TABANLI GÖRÜNTÜ SINIFLANDIRMA TEKNİĞİ İLE X-RAY TARAMA GÖRÜNTÜLERİNDEN USB BELLEK İÇERENLERİN SINIFLANDIRILMASI

Ali HACIHAMZAOĞLU

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"ELEKTRONİK YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 15/ 08 / 2023
Tezin Savunma Tarihi : 11 / 09 / 2023

Tez Danışmanı : Prof. Dr. İsmail Hakkı ÇAVDAR

Trabzon 2023

ÖNSÖZ

“Yapay Sinir Ağı Tabanlı Görüntü Sınıflandırma Tekniği ile X-ray Tarama Görüntülerinden USB Bellek İçerenlerin Sınıflandırılması” isimli yüksek lisans tez çalışmamda danışmanım olarak beni yönlendiren, bana fikirleriyle vizyon katan, aldığım yüksek lisans eğitimi boyunca desteğini her zaman hissettiğim çok değerli saygıdeğer hocam Sayın Prof. Dr. İsmail Hakkı ÇAVDAR’ a teşekkür ederim.

Tez çalışmam boyunca, desteklerini hiç esirgemeyen, önerileri ve bana kattıkları vizyon sayesinde tezime büyük katkıları bulunan saygıdeğer hocalarım Sayın Doç. Dr. Volkan YILMAZ ve Sayın Dr. Öğr. Üyesi Çiğdem ŞERİFOĞLU YILMAZ’ a teşekkür ederim.

Yapay zekâ uygulamaları ve derin öğrenme algoritmaları konusunda beni yönlendiren, teşvik eden, maddi ve manevi desteğini her zaman hissettiğim dayım Yüksel PENEKLİ’ ye teşekkür ederim.

Son olarak, her zaman yanımda olan, her kararımda arkamda olup bilgi ve desteklerinden faydalandığım maddi ve manevi desteklerini hiçbir zaman esirgemeyen ve bugünlere gelmemde en büyük paya sahip olan annem Meryem HACIHAMZAOĞLU, babam Bülent HACIHAMZAOĞLU’ na, kardeşim Atakan HACIHAMZAOĞLU’ na ve eşim Didem HACIHAMZAOĞLU’ na teşekkür eder, minnettarlığımı sunarım.

Bu tez çalışmasının benzer konulardaki çalışmalara katkıda bulunmasını temenni ederim.

Ali HACIHAMZAOĞLU

Trabzon 2023

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “YAPAY SİNİR AĞI TABANLI GÖRÜNTÜ SINIFLANDIRMA TEKNİĞİ İLE X-RAY TARAMA GÖRÜNTÜLERİNDEN USB BELLEK İÇERENLERİN SINIFLANDIRILMASI” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. İsmail Hakkı ÇAVDAR’ ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 11/09/2023

ALİ HACIHAMZAOĞLU

İÇİNDEKİLER

Sayfa No

ÖZET

Yüksek Lisans Tezi

YAPAY SİNİR AĞI TABANLI GÖRÜNTÜ SINIFLANDIRMA TEKNİĞİ İLE X-RAY TARAMA GÖRÜNTÜLERİNDEN USB BELLEK İÇERENLERİN SINIFLANDIRILMASI

Ali HACIHAMZAOĞLU

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik ve Elektronik Mühendisliği Anabilim Dalı (Elektronik)
Danışman: Prof. Dr. İsmail Hakkı Çavdar
2023, 135 Sayfa

X-ray tarama teknolojisinin gelişmesi ve yaygınlaşması sonucunda, üretilen görüntülerin sayısı da aynı oranda artmaktadır. X-ray tarama sistemleri tıptan endüstriye birçok alanda kullanıldığı gibi güvenlik alanında sıklıkla tercih edilmektedir. Gelişen teknoloji ile yüksek çözünürlüklü görüntü üretme kapasitesine sahip yeni nesil X-ray cihazları güvenlik alanında, potansiyel tehditlerin tespiti konusunda hayati önem taşımaktadır. Ancak kullanım esnasında hızlı tarama yapabilen bu cihazların ürettiği görüntülerinde kullanıcılar tarafından hızlı bir şekilde analiz edilmesi gerekmektedir. Özellikle veri güvenliğini tehdit eden USB Bellek gibi küçük cihazların hızlı bir şekilde X-ray görüntülerinden tespiti bir sorun oluşturmaktadır. Bu çalışma, derin öğrenme tabanlı sınıflandırma yöntemleri ile tarama görüntülerinde USB bellek olup olmadığını tespit etmeyi amaçlamaktadır. Bu kapsamda, X-Ray tarama sistemleriyle 1217 adet USB Bellek içeren ve içermeyen veri oluşturulmuştur. Oluşturulan veri seti Evrişimsel Sinir Ağları (ESA) tabanlı derin öğrenme mimarisine sahip 8 farklı model ile eğitilmiştir. Yapılan eğitim sonucunda (%93) başarı oranı ile en yüksek genel doğruluk değerine ResNet50V2 modeli ile ulaşılırken, ResNet50 modeli ile en düşük (%67) genel doğruluk değeri elde edilmiştir.

Anahtar Sözcükler: X-ray, Tarama Sistemleri, Yapay Zekâ, Yapay Sinir Ağları, USB bellek, ESA

SUMMARY

Master Thesis

CLASSIFICATION OF X-RAY SCAN IMAGES CONTAINING USB MEMORY BY ARTIFICIAL NEURAL NETWORK-BASED IMAGE CLASSIFICATION TECHNIQUE

Ali HACIHAMZAOĞLU

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Electrical ve Electronics Engineering Graduate Program (Electronics)
Supervisor: Prof. Dr. İsmail Hakkı Çavdar
2023, 135 Pages

As a result of the development and widespread use of X-ray scanning technology, the number of images produced is increasing at the same rate. X-ray scanning systems are used in many fields from medicine to industry and are frequently preferred in the field of security. New generation X-ray devices, which have the capacity to produce high resolution images with the developing technology, are of vital importance in the detection of potential threats in the field of security. However, the images produced by these devices, which can scan quickly during use, need to be quickly analyzed by the users. The rapid detection of small devices such as USB memory sticks, which threaten data security, from X-ray images poses a problem. This study aims to detect whether there is USB memory in scan images with deep learning-based classification methods. In this context, data with or without 1217 USB sticks were created with X-Ray scanning systems. The created data set is trained with 8 different models with Convolutional Neural Networks (CNN) based deep learning architecture. As a result of the training, the ResNet50V2 model achieved the highest overall accuracy with a success rate (93%), while the lowest (67%) overall accuracy value was obtained with the ResNet50 model.

Keywords: X-ray, Scanning Systems, Artificial Intelligence, Artificial Neural Networks, USB Memory, CNN

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

1. GENEL BİLGİLER

1.1. Giriş

Röntgen ışınları ile tarama teknolojisinin gelişmesi ile X-ray bagaj tarama sistemlerinin kullanımı da bu doğrultuda oldukça artmıştır. Havalimanları, gümrük kapıları, otogarlar ve çeşitli güvenlik alanlarında kullanılan X-ray tarama cihazları tehlikeli nesnelerin tespit edilmesinde önemli bir rol oynamaktadır. Bu sistemler, bagaj ve eşyaların içeriğini görüntülemek ve potansiyel tehlikeli veya yasaklı maddeleri tespit etmek için X ışınlarını kullanmaktadır (Mamchur, D., Peksa, J., Le Clainche, S., & Vinuesa, R.,2022). Eşyalar, X-ray bagaj tarama sistemlerinde genel olarak bir hareketli ray üzerinde kaydırılırken X ışını emilimine maruz bırakılır. Daha sonra bu emilim verileri, bir görüntü oluşturmak amacıyla algılayıcılar tarafından yakalanır ve dijital verilere dönüştürülüp bilgisayar vasıtasıyla görüntüleme ünitelerine aktarılır. X-ray bagaj tarama sistemlerini oluşturan genel olarak ana temel bileşenler şunlardır:

- X-ray Kaynağı: Yüksek voltajlar aracılığı ile X ışını tüpüyle X ışını üreten bölüm (Prabhu, S., Naveen, D. K., Bangera, S., & Bhat, B. S.,2020).
- Detektör: X-ray ışınları ile etkileşime giren nesnelerin desenlerini toplayan birim.

İşlemci ve Görüntüleme Birimi: Detektörlerden elde edilen verileri işleyen ve görüntüleyen çevirici bilgisayar sistemleri.

- Taşıyıcı Bant: Taranması gereken eşyaların X-ray tarama sisteminin içinden geçmesini sağlayan bant sistemi.

Günümüzde X-Ray tarama sistemleri çoğunlukla tarama sonuçlarını RGB formatında vermektedir. Bununla beraber ortaya çıkan verinin analizi kolaylaşmakta ve tehlikeli nesnelerin tespiti daha çabuk olmaktadır. Ayrıca oluşan görüntüler çözünürlük açısından son yıllarda daha iyi bir sonuç vermektedir.

X-Ray tarama sistemlerinin yaygın kullanımı ile oluşan görüntü sayısı paralel olarak artmaktadır. Buna bağlı olarak incelenmesi gereken büyük veri kümeleri ortaya çıkmaktadır. Bu büyük veri kümelerinin getirdiği analiz ve tespit zorluklarına derin öğrenme ve yapay zekâ teknolojisi başarılı çözümler sunmaktadır. Görüntü sınıflandırma ve nesne tespiti konusunda sıklıkla kullanılan derin öğrenme teknolojisi aynı zamanda X-ray görüntülerinin

sınıflandırılması kısmında da yeni çözümler sunmuştur (Abbas, A., Abdelsamea, M. M., & Gaber, M. M.,2021).

Geçmişi daha önceki yıllara dayanan derin öğrenme ve yapay zekâ teknolojisi, özellikle son yıllarda bilgisayar donanım teknolojilerinin ciddi seviyede gelişmesiyle önemli ölçüde öğrenim sürelerinin kısalması sonucunda büyük değer kazanmıştır. Birçok bilimsel çalışma da yenilik sağlayan derin öğrenme algoritmaları popüler olarak kullanılmaktadır. Sağlık alanında sıklıkla kullanılan X-Ray görüntüleme teknolojilerinde başarılı sonuçlar veren derin öğrenme ve görüntü sınıflandırma algoritmaları COVID-19 olan bir bireyin X-Ray göğüs tarama görüntüsünü, COVID-19 olmamış bireyin X-Ray göğüs tarama görüntüsünden ayırabilmektedir (Toğaçar, M., Ergen, B., & Cömert, Z.,2020). Derin öğrenme algoritmaları insana ihtiyaç duymadan X-Ray görüntüleri üzerinde otomatik olarak tespit ve sınıflandırma yapabilmektedir ve bununla beraber başarılı sonuçlar elde edilmektedir (Nair, R., Vishwakarma, S., Soni, M., Patel, T., & Joshi, S.,2022). X-ray görüntüleri kullanılarak Mask R CNN yaklaşımı (Zhang, J., Song, X., Feng, J., & Fei, J.,2021), ESA yaklaşımı (Akcay, S., & Breckon, T. P.,2017), Segmentasyon çalışmaları (Griffin, T., Cao, Y., Liu, B., & Brunette, M. J.,2020) vb. gibi derin öğrenme çalışmalarına örnek olarak gösterilebilmektedir.

1.2. Çalışmanın Amacı

X-Ray tarama sistemi teknolojisinin gelişmesi ve yaygınlaşması ile tarama sonucu incelenen görüntü sayısı artmaktadır. Ancak yoğun kargo veya bagaj geçişlerinin olduğu zaman aralıklarında yasaklı veya tehlikeli maddelerin tespiti kullanıcı açısından zorlaşmaktadır. Kısa sürede her görüntüyü ayrıntılı inceleyip yasak nesne tespiti yapmak kullanıcı açısından zor olduğu için bir güvenlik açığı oluşmaktadır. Çoğu teknolojiyi takip eden X-ray bagaj tarama sistemi üreticileri bu sebep ile, gelişmiş yapay sinir ağı ve görüntü işleme teknolojileri kullanarak, potansiyel tehditleri otomatik olarak tespit edebilmek ve X-ray kullanıcılarına tehlikeyi önceden haber vermek istemektedir. Nesne tespiti alanındaki literatür incelendiğinde, çeşitli yaklaşımların bulunduğu görülmektedir (Saavedra, D., Banerjee, S., & Mery, D.,2021). X-Ray bagaj taramalarında tehlikeli nesnelerin tespiti için derin öğrenme tabanlı YOLO (You Only Look Once) algoritması ve ESA kullanılarak silah, bıçak ve jilet vb. gibi tehlikeli aletlerin sınıflandırmasını sağlayan çalışmalar olmuştur (Jain, D. K.,2019). Silah, bıçak, bomba vb. gibi bilinen tehlikeli nesnelerin tespiti ile alakalı

literatürde derin öğrenme tabanlı çalışmalar yapılmıştır (Ponnusamy, V., Marur, D. R., Dhanaskodi, D., & Palaniappan, T.,2021). Fakat günümüzde çeşitli kurum ve kuruluşlar tehlike oluşturabilecek nesneler ile kalmayıp gizli verilerin kaybını önlemek için sıkı güvenlik önlemlerine de ihtiyaç duymaktadır. USB bellekler, veri ihlali potansiyeline ve casus yazılımların (virüs, solucan, Trojan vb.) bilgisayarlara veya ağ bağlantılara bulaşmasında rol sahibi olan önemli bir tehdit oluştururlar.

Bu çalışma, bilgisayarlı görü çalışmaları ve yapay sinir ağı algoritmalarının kullanımıyla, X-ray tarama görüntülerinde USB belleklerin tespitini kullanıcı görüsü olmadan makine görüsü ile sınıflandırmayı amaçlamaktadır. Bagaj X-Ray görüntüleme sistemi ile elde edilen 1217 adet USB bellek içeren ve içermeyen veri kümesi, 8 farklı Evrişimsel Sinir Ağı (ESA) tabanlı derin öğrenme mimarisine farklı kombinasyonlar ve teknikler ile öğretilip test edilmiştir. Yapılan çalışmaların ağ mimarisinin ilk kısmını oluşturan Evrişim katmanında VGG16, VGG19, Resnet50v2, MobileNetV2, InceptionResNetV2, InceptionV3, ResNetRS50, Xception Evrişimsel Sinir Ağı (ESA) tabanlı derin öğrenme mimarileri kullanılmıştır.

1.3. X-Ray Görüntülerinde Sınıflandırma ve Önemi

X-Ray tarama sistemlerinden çıkan görüntülerin sınıflandırılması hem endüstri alanında hem tıp alanında çok önemlidir. Özellikle tıp alanın hastalığın tanısında ve teşhisinde önemli payı olan görüntü sınıflandırma işlemi hastalıkların erken dönemde tespitine bu sayede tedavinin erkenden başlamasına yardımcı olmaktadır. Güvenlik uygulamalarında da ciddi rol oynayan görüntü sınıflandırma metodu tehlikeli nesnelerin görüntüdeki yerinin tespiti ve tehlikeli nesne içeren görüntüyü sınıflandırma işlemleri ile güvenlik hassasiyeti olan kuruluşları rahatlatmaktadır. Zamandan ve işgücünden tasarruf etmeyi sağlayan bu yöntem ile alakalı literatürde ciddi çalışmalar yapılmıştır. Derin öğrenme yöntemleriyle zatürre geçiren bir hastanın X-ray görüntüsünden yüksek başarı skorlarına sahip bölgesel tespit yapılabilmektedir (Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., & Acharya, U. R.,2020). Aynı zamanda eşyaların tarandığı bagaj X-ray sistemlerinde de tehlikeli nesnelerin olduğu görüntülerin sınıflandırması işlemi sonucunda skorları başarılı olan derin öğrenme yaklaşımları mevcuttur (Gaus, Y. F. A., Bhowmik, N., & Breckon, T. P.,2019).

1.4. Metodoloji

Yapılan çalışmada kullanılacak yöntemler sıralı bir şekilde aşağıda verilmiştir.

- Derin öğrenme yöntemlerinden olan Evrişimsel Sinir Ağları (ESA) yaklaşımı için gerekli araştırmalar yapılıp, X-ray tarama sistemlerinin ürettiği görüntülerin sınıflandırılması amacıyla kullanılıp kullanılamayacağı araştırılmıştır.
- Derin öğrenme algoritmalarını eğitmek amacıyla kullanmak üzere veri setleri elde edilmiştir. Veri setleri Rapiscan 620DV X-ray tarama cihazıyla üretilmiştir.
- 1217 adet veri 10 tip USB Bellek ve çeşitli materyaller ile birlikte taranarak 807 âdeti USB Bellek içerecek şekilde, 410 âdeti USB bellek içermeyecek şekilde ayrılmıştır. Görüntüler ortalama 450x900 çözünürlükte RGB formatta üretilmiştir. RGB formatta üretilen görüntüler olan Evrişimsel Sinir Ağları (ESA) yaklaşımı için daha iyi eğitim ve buna bağlı sonuç skoru alma imkânı sunmaktadır.
- Verilerin eğitim için ayrılan 1003 âdeti veri artırımı yöntemi ile çeşitli şekillerde çoğaltılmıştır. Bu çoğaltma esnasında görüntülerin bir kısmı kendi etrafında belli açı değerlerinde döndürülmüş, bir kısmına yakınlaşılmıştır. Veri artırımı yöntemi ile normal veriler sayesinde eğitilen modellerin başarı ve hata skorları, artırılmış veriler tarafından eğitilmiş modellerin başarı ve hata skorları ile karşılaştırılmıştır. Ayrıca veri artırımı yöntemi, başka bir öğrenim metodu olan öğrenim aktarımında *(Transfer Learning)* kullanılmak amacıyla da uygulanmıştır.
- Üretilen görüntüler modele eğitim ve test amacıyla verilmeden önce çözünürlük değerleri 224x224 seviyesine düşürülmüştür. Bu işlem sayesinde modelin parametre sayısı düşürülüp öğrenim aşamasının daha hızlı bir şekilde gerçekleşmesi amaçlanmıştır.
- USB Bellek içeren görüntüleri içermeyen görüntülerden ayırma işlemini gerçekleştirecek olan yapay zekâ tabanlı model mimarisi hazırlanmıştır.
- Oluşturulacak olan mimari, literatürde başarılı sonuçlar veren 8 farklı ESA mimarisi temel alınarak oluşturulmuştur.
- Oluşturulan modellerin hazırlanan veriler ile eğitimleri yapılmıştır. Eğitim sonucu ağırlıkları değişen modeller test verilerine tabii tutulup oluşan grafikler incelenmiştir. Oluşan eğitim-test doğruluk ve hata sonuçların, adım sayısı ile karşılaştıran grafikler modelin eğitim esnasında aşırı uydurmaya maruz kalıp kalmadığını tespit etmek için önem arz etmektedir.

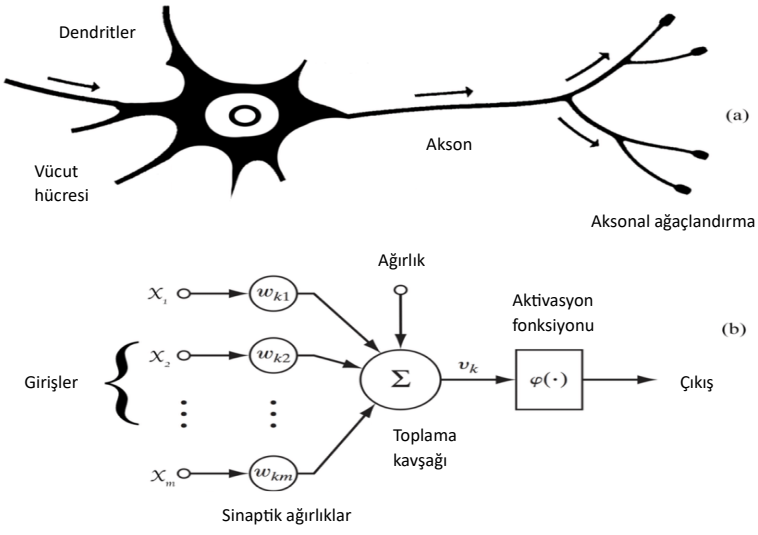
- Ağırlıkları arttırılmamış eğitim verilerine göre belirlenen modeller, öğrenim aktarımı yönteminin getirdiği olumlu sonuçlar test edilmek üzere artırılmış veri ile tekrar eğitilip sonuç grafikleri tekrar incelenmiştir.
- Artırılmamış veriler, artırılmış veriler ile eğitilen ve öğrenim aktarımı yöntemi ile güncellenen 3 adet ağırlıkları farklı aynı Evrişimsel Sinir Ağları (ESA) mimarisine sahip modellerin eğitim ve test skorları kaydedilmiştir.
- Eğitim ve test skorları ile modellere ait genel doğruluk, kesinlik, duyarlılık ve F1 skorları üretilmiş ve bu skorlara göre oluşturulan ağırlıkları yeni belirlenmiş modellerin başarıları değerlendirilmiştir.

1.5. Yapay Sinir Ağları

Yapay Sinir Ağları (YSA), biyolojik görsel korteks sistemlerinden esinlenerek tasarlanmış matematiksel modellemelerdir. Bu modellemeler ile büyük miktarda veriyi işleyebilme, karmaşık örüntüleri tanıma ve anlama yeteneğine sahip olan yapay zekâ teknolojisinin temelini oluşmuştur. Yapay sinir ağları, öğrenme ve genelleştirme yapabilme yeteneğine sahip, esnek ve uyarlanabilir algoritmalar olarak tanımlanmaktadır.

Yapay sinir ağlarını oluşturan temek birim "nöron" olarak isimlendirilen mantıksal modellerdir. Bu nöronlar, veriler aracılığıyla sonuçlar üretebilen ve bu işlem uygulanırken ağırlık ve eğim gibi parametrelere sahip olan matematiksel işlemler gerçekleştirir.

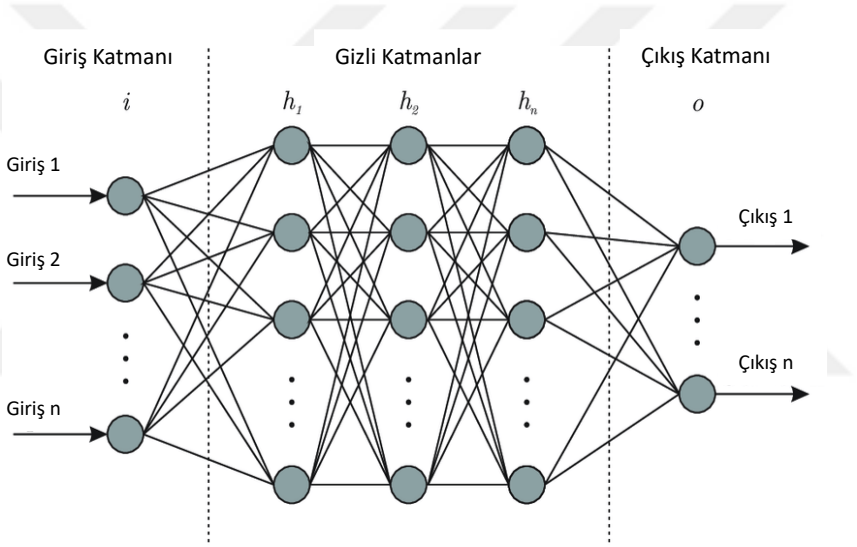
Yapay sinir ağları, katmanlar halinde öğrenme işlemini gerçekleştiren nöronlardan oluşur.



Şekil 1. Yapay sinir hücresi (Akgün, E., & Demir, M., 2018)

Temelde üç çeşit katmandan oluşmaktadır:

- Giriş Katmanı *(Input Layer)*: Giriş verilerinin ağa girdiği ve işleme başlandığını ifade eden katmandır. Bu katmana gelen verinin her bir özelliğine göre nöron belirlenir.
- Gizli Katmanlar *(Hidden Layers)*: Bir veya birden daha çok katman içerirler. Bu katmanlarda, giriş katmanından gelen verilerden anlamlı sonuçlar üretebilmek için öğrenme işlemi uygulanır. Gizli katmanlar aynı zamanda ağın karmaşık yapısını ve gücünü belirler.
- Çıkış Katmanı *(Output Layer)*: Model mimarisinin son katmanıdır ve çoğunlukla verilen veriler sonucunda anlamlı çıktıları üretir.



Şekil 2. Yapay sinir ağı mimarisi (Bre, F., Gimenez, J. M., & Fachinotti, V. D., 2018)

Eğitim süreci boyunca Yapay sinir ağları eğitilir. Eğitim verileri ağa sokulur ve ağın ürettiği sonuçlar gerçek sonuçlar ile karşılaştırılır. Ardından, ağırlıklar ve eğimler, hatanın düşürülmesi için geriye doğru yayılma *(Back-propagation)* metodu ile tekrar ayarlanır. Bu işlem süreci her adım tekrarlanarak ağın performansının geliştirilmesi amaçlanmaktadır.

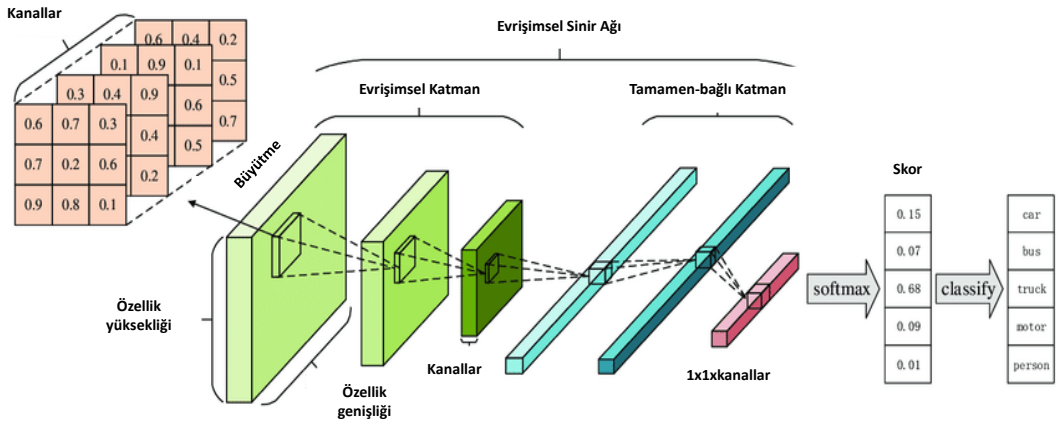
Yapay sinir ağları sayesinde, geleneksel olarak kullanılan regresyon ve istatiksel yöntemlere göre çok daha başarılı sonuçlar alınabilmektedir (Dave ve Dutta, 2014).

Yapay sinir ağları, görsel tanıma, doğal dil tanıma, ses tanıma, nesne tespiti ve daha birçok alanda kayda değer sonuçlar veren bir yapay zekâ teknolojisidir. Özellikle derin öğrenme ile birleştirildiğinde, büyük veri kümelerinde başarılı skorlar elde edilebilir ve

çeşitli karmaşık görevler başarıyla gerçekleştirilebilir. Birçok araştırmacı sınıflandırma problemlerini çözümlemek için makine öğrenimi yöntemlerini kullanmaktadır (Das ve Behera, 2017).

1.5.1. Evrişimsel Sinir Ağları

Evrişimsel sinir ağları, görüntü sınıflandırma problemleri ve nesne tespiti uygulamalarında çok sık kullanılan ve iyi sonuçlar üretebilen derin öğrenme mimarileridir. Evrişimsel sinir ağlarına gelen verinin özelliklerinin ayrıca girilmesine gerek yoktur. Yapay sinir ağlarının ağırlıklarının belirlenmesi gibi evrişimsel sinir ağları da birtakım ağırlıklara sahiptir ve bu ağırlıkları çekirdeklerinde tutmaktadır. Taranan görüntüler belirlenen filtre sayısı ile evrişim katmanlarında tarandıktan sonra elde edilen değerler havuzlama katmanına aktarılır. Bu işlem sırasında evrişimsel sinir ağlarının ağırlıkları güncellenmektedir. Gelişen teknolojiye bağlı azalan eğitim süreleri ile evrişimsel sinir ağlarının kullanımı artmıştır ve buna bağlı gelişmiştir.



Şekil 3. Evrimsel sinir ağı mimarisi (Kang, X., Song, B., & Sun, F., 2019)

Evrişim Katmanları

Evrişim katmanları üzerlerinde oluşturulan çekirdekleri kullanarak evrişim yaparlar. Bu evrişim işleminin amacı giren verinin bir özellik haritasını çıkarmaktır. Ana iskeleti oluşturan evrişim hesapları tek boyutlu, iki boyutlu ve üç boyutlu olacak şekilde yapılabilmektedir. Formülleri ise aşağıdaki şekildedir;

$$w_{ij}^{x} = f\left(b_{ij} + \sum_{P=0}^{P_i-1} \sum_{s=0}^{S_i-1} v_{ij}^{s}\, w_{(i-1)p}^{(x+s)}\right) \quad (1)$$

$$w_{ij}^{xy} = f\left(b_{ij} + \sum_{P=0}^{P_i-1} \sum_{s=0}^{S_i-1} \sum_{r=0}^{R_i-1} v_{ij}^{sr}\, w_{(i-1)p}^{(x+s)(y+r)}\right) \quad (2)$$

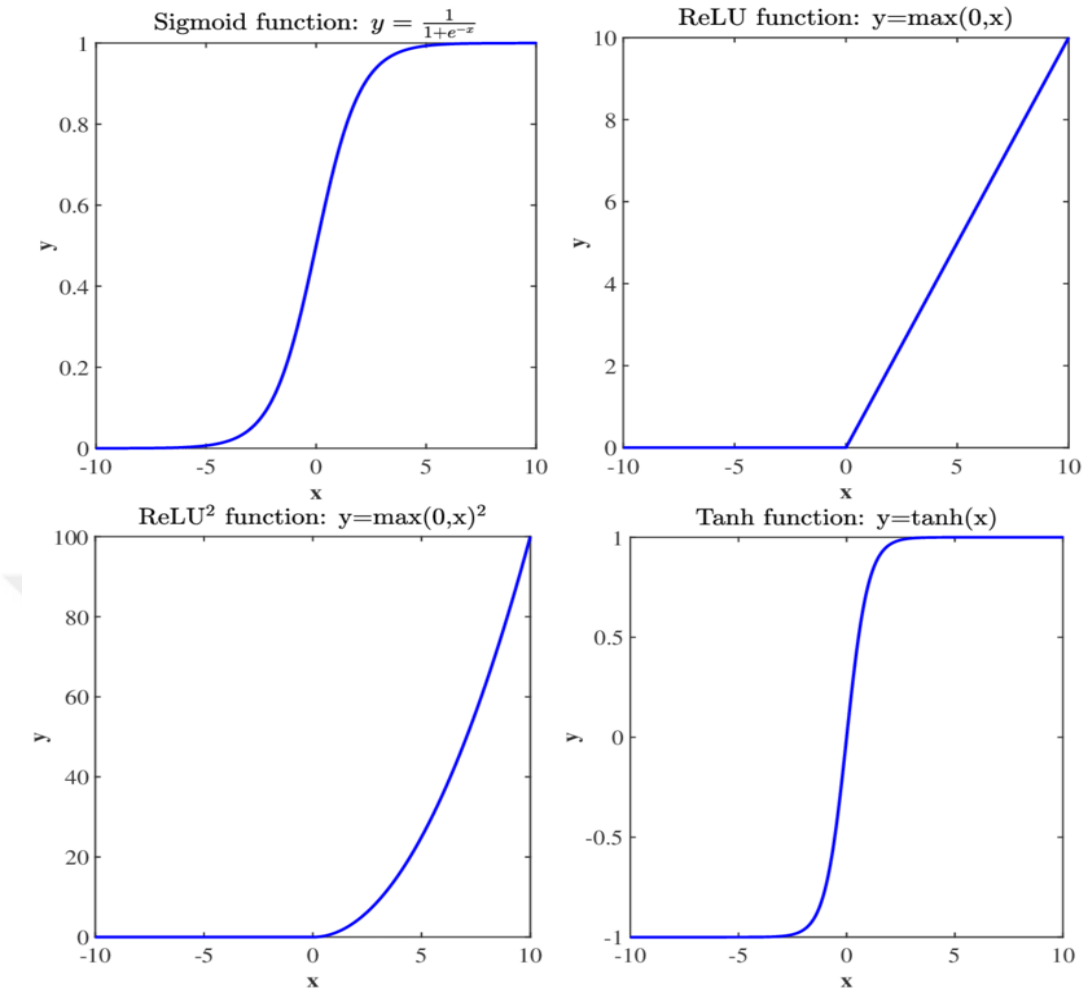
$$w_{ij}^{xyz} = f\left(b_{ij} + \sum_{P=0}^{P_i-1} \sum_{s=0}^{S_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} v_{ijp}^{qrs}\, w_{(i-1)p}^{(x+q)(y+r)(z+s)}\right) \quad (3)$$

Yukarıda görülen formüllerde, w özellik haritalarından alınan çıktıyı, S, Q, R ise boyutsal ve RGB çekirdek özelliklerini, (s, q, r) çekirdek indekslerini ve (x, y, z) 2 boyut ve 1 RGB kanal verisinin indeksini ifade etmektedir. i, j, p de sırayla giren katman, çıkan katman ve özellik harita indeksini göstermektedir. Ağırlık indeksi b ile ifade ediliştir. F fonksiyonu ReLU fonksiyonunu temsil etmektedir.

Evrişim işleminde girdi olarak gelen görüntü belirlenen sayıdaki katman ve katmanlardaki filtreler tarafından sol üstten başlanarak taranır. Bu tarama sonucunda mimarideki nöronların ağırlıkları belirlenmektedir. Girdiye uygun filtreler ve buna bağlı ağırlıklar ESA tarafından otomatik bir şekilde bulunmaktadır (Gad vd., 2018). Bu tez çalışmasının evrişimsel sinir ağı kısmında 8 farklı model kullanılmıştır.

Doğrusal Olmayan Katman *(Non-Linearity Layer)*

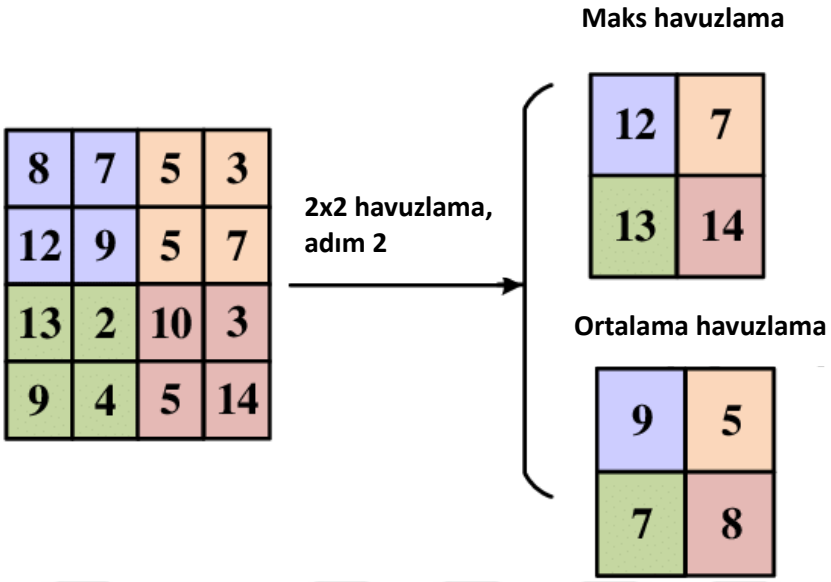
Giren verilerin evrişim katmanından çıktıktan sonra çıktılarına bir değer verebilmek için doğrusal olmayan fonksiyon katmanı kullanılmaktadır. Genellikle ReLU aktivasyon fonksiyonu tercih edilmektedir. Negatif değer almayan ReLU aktivasyon fonksiyonunda negatif değerler 0 olarak ayarlanmaktadır. Negatif değerlerin sıfır olarak ayarlanması ağa daha hızlı çalışma imkânı sunmaktadır (Venkatesan vd., 2017).



Şekil 4. Doğrusal olmayan aktivasyon fonksiyonları (Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., ... & Rabczuk, T., 2020)

Havuzlama Katmanı *(Pooling Layer)*

Havuzlama işlemi, genel olarak giren verinin benzer özellik haritasını anlamlı olarak birleştirerek daha sonraki katmana daha küçük boyutlu bir veri girmesini amaçlamaktadır. Ortalama ve maksimum havuzlama katmanı olarak kullanılan bu işlem evrişim işleminin maliyetini düşürmeyi hedeflemektedir.



Şekil 5. Ortalama ve maksimum havuzlama işlemi örneği (Yingge, H., Ali, I., & Lee, K. Y., 2020)

Ağ mimarilerinde ise maksimum havuzlama işlemi, diğer havuzlama işlemlerinden daha iyi sonuçlar verdiği için genel olarak tercih edilmektedir. (Lee vd., 2016).

Tamamen Bağlı Katman *(Fully-Connected Layer)*

Evrişimsel katmandaki çıktılardan anlamlı verilerin öğrenilme işleminin gerçekleştiği kısımdır. Giriş kısmını yani evrişimsel kısmı çıkış katmanına bağlayan çok katmanlı ve her katmanında çok sayıda nöron içeren yapıdır.

$$y_i = f(W.u_i) + b^h \quad (4)$$

Formülde; u_i : giren veri, W: nöron ile alakalı ağırlık matrisi, b: ağırlık değeri, f: aktivasyon fonksiyonudur (Bouvrie, 2006).

Ayrıca çalışmada bu katmanlar arasına aşırı uydurma *(over-fitting)* durumunu önlemek için Drop-out katmanları yerleştirilmiştir. Drop-out katmanlarının belli oranlarda rastgele bağlantıları kaldırmasıyla modelin gerçek veriler ile daha iyi bir sonuç vermesi amaçlanmıştır. Dinamik Drop-out katmanlarının öğrenim işleminde uygulanan adım sayısını düşürüp daha hızlı sonuçlar verdiği literatürde gözlenmiştir. (Teso Fernández de Betoño, A.,

Zulueta Guerrero, E., Cabezas Olivenza, M., Fernández Gámiz, U., & Botana Martínez de Ibarreta, C.,2023)

Çıkış Katmanı *(Output Layer)*

Yapı itibariyle tamamen bağlı katmana benzer yapıda olan bu katmanda modelin ürettiği sonuçlar sınıflandırılıp kullanıcıya sunulmaktadır. Bu katmanda, sınıflandırma işleminde kullanılan sınıf sayısı kadar nöron kullanılmaktadır. Çıkış katmanındaki her nöron, belli sınıflara karşılık gelecek şekilde sonuçlar üretmektedir ve bu sonuçlardan maksimum olanı tercih edilmektedir (Morchhale, 2016).

1.5.2. Hiper Parametrelerin Seçimi

Derin öğrenme modelleri çok sayıda hiperparametre ile çalışır. Derin öğrenme algoritmaları karmaşık yapılara sahip olduğundan dolayı hiperparametrelerin doğru belirlenmesi modelin performansı açısından çok önemli bir rol oynamaktadır. Bu parametrelere eğitim ve test süreçlerini kontrol eden ayarlar olarak düşünülebilir. Hiperparametreler modele, modelin mimarisine, eğitim verisinin çeşitliliğine, eğitim verisinin büyüklüğüne ve daha birçok parametreye bağlı değişebilmektedir. Hiperparametreler deneme-yanılma, katlamalı çapraz doğrulama *(k-fold cross-validation)*, rastgele arama, grid arama ve bayes optimal arama yöntemleri ile belirlenebilmektedir. En iyi hiperparametre seçimi modele ve veri kümesine en uygun parametreleri kombinasyonlamaktır. Yığın boyutu, adım sayısı, optimizasyon fonksiyon parametreleri, katman sayısı, nöron tipi vb. değişkenler önemli hiperparametrelere örnek gösterilebilmektedir.

1.5.3. Optimizasyon Fonksiyonu

Yapay zekâ uygulamalarında optimizasyon fonksiyonunun görevi model ağırlıklarını ve öğrenme hızını ayarlayıp kayıp fonksiyonunu en düşüğe indirmektir. Kayıp fonksiyonunun büyük değer vermesi modelin istenilen seviyeden uzaklaştığı anlamına gelir. Optimizasyon fonksiyonu ise gradyan yöntemi ile modelin her bir adımda ağırlıklarının güncellenmesine yardımcı olur. Derin öğrenme algoritmalarında kullanılan bazı optimizasyon fonksiyonları şunlardır:

- Stokastik Gradyan İnişi: Büyük veri kümelerinde kullanılması eğitimi hızlandırsa da bazı durumlar için düzensiz davranabilmektedir.
- Mini-Batch Gradyan İnişi: Veri kümesini küçük gruplara ayırarak işlenmesini sağlar.
- RMSprop *(Root Mean Square Propagation)*: Gradyanın karesinin ortalamasını alarak öğrenme hızını ayarlar.
- Adam: Adam algoritması yaygın olarak kullanılır. Her adımda gradyanları kullanarak ağırlıkları günceller.

Yapılan çalışmada Adam optimizasyon algoritması tercih edilmiştir.

1.6. Nesne Tespiti Amaçlı Üretilen ve Çalışmada Kullanılan ESA Modelleri

Gelişen derin öğrenme algoritmalarına bağlı olarak nesne tespiti çalışmalarında büyük mesafe katedilmiştir. Bu çalışmada, oluşturulan modelin nesne tespiti için ana gövdeyi oluşturan ESA katmanında kullanılan 8 farklı model sırayla şöyledir:

- VGG16, VGG19, ResNet50V2, MobileNetV2, InceptionResNetV2, InceptionV3, ResNet50, Xception

1.6.1. VGG16

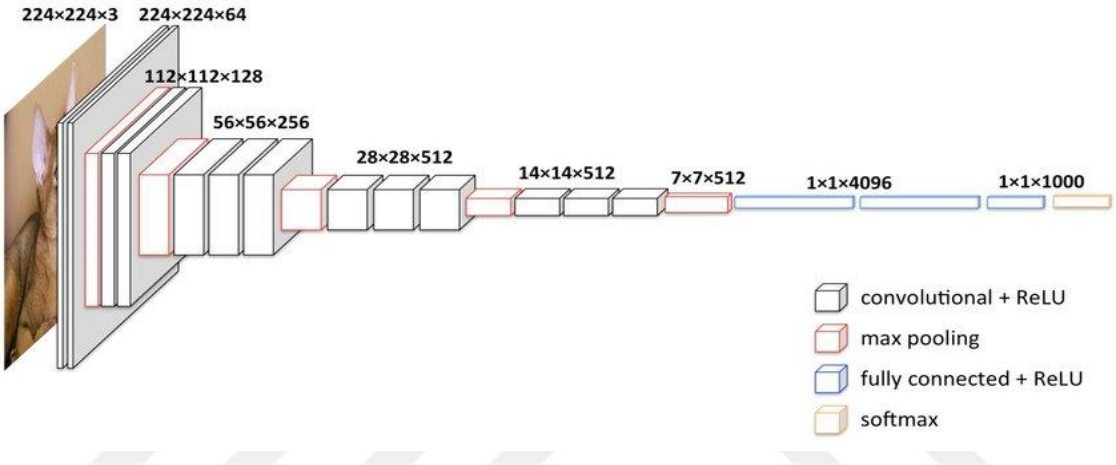
VGG-16 (Visual Geometry Group 16) modeli, 2014 senesinde Oxford Üniversitesi'ndeki Visual Geometry Group (VGG) tarafından geliştirilen derin öğrenme mimarisidir. Model, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasına birden fazla model ile katılmış ve yarışmadaki en iyi sonuçları alarak büyük bir başarı elde etmiştir.

Model, genel olarak basit ve simetrik bir yapıya sahiptir, bu sebeple anlaşılması ve uygulanması çok basittir. VGG-16, 16 adet katman (13 adet evrişimsel katman ve 3 adet tam bağlantılı katman) içerir.

VGG-16'nın yapısını özetleyecek olursak:

- Evrişimsel Katmanlar: 3x3 boyutunda filtrelerle çeşitli evrişimsel katmanlar kullanır. Sırayla 64, 128, 256, 512 ve 512 adet filtre içeren beş evrişimsel katman bulunmaktadır.

- Aktivasyon Fonksiyonu: Her evrişimsel katmanın ardından ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu kullanılır, bu sayede ağa daha derin öğrenme imkânı sağlanmaktadır.
- Havuzlama Katmanları: Modelde 2x2 boyutunda maksimum havuzlama *(max-pooling)* katmanları kullanılarak boyut düşürme işlemi gerçekleştirilir.
- Tam Bağlantılı Katmanlar: Son evrişimsel katmandan sonra üç adet tam bağlantılı katman bulunur. Bu katmanlar, sınıflandırma işlemi için kullanılır.



Şekil 6. VGG16 ağ mimarisi (Shi, B., Hou, R., Mazurowski, M. A., Grimm, L. J., Ren, Y., Marks, J. R., ... & Lo, J. Y., 2018)

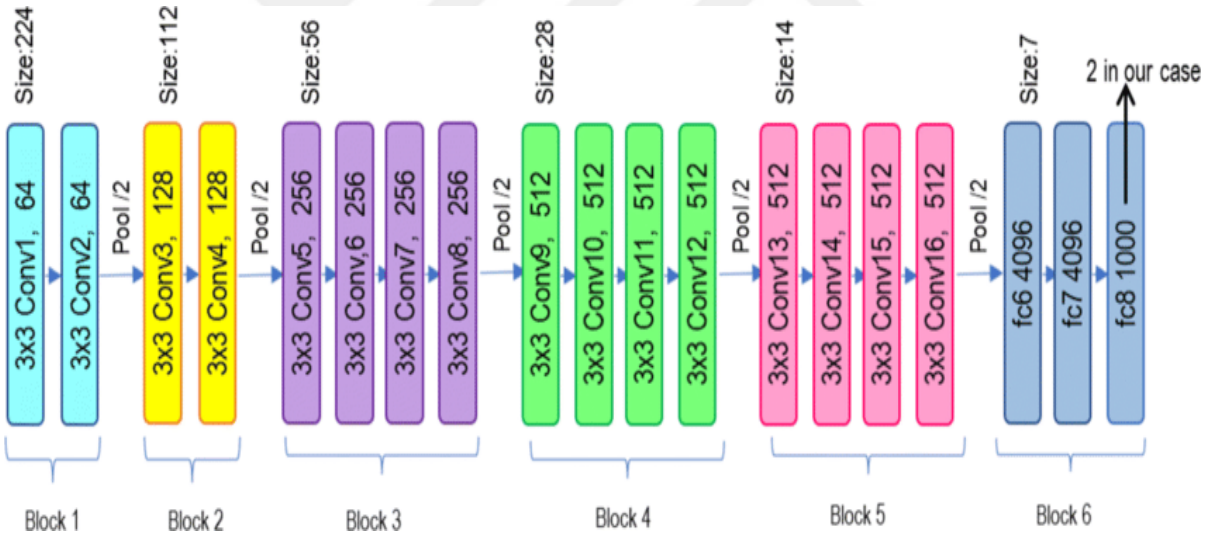
1.6.2. VGG19

VGG-19, VGG-16 modelinin yükseltilmiş daha yeni bir versiyonudur ve Visual Geometry Group (VGG) tarafından 2014 yılında geliştirilmiştir. VGG-19, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasına katılan modeller arasında yer almış ve VGG-16'dan daha derin bir yapıya sahip olarak tanınmıştır.

VGG-19'un temel özellikleri VGG-16 ile genel olarak aynıdır fakat farklı olarak daha fazla evrişimsel katmana ve daha fazla tam bağlı katmanlara sahiptir. Temel özelliklerini sıralayacak olursak:

- Katman Yapısı: VGG-19, 19 adet katman içerir. 19 adet katman 16 adet evrişimsel katman ve 3 adet tam bağlantılı *(fully connected)* katmandan oluşur.

- Evrişimsel Katmanlar: VGG-19, ardışık olarak 3x3 boyutunda filtrelerle 16 adet evrişimsel katman kullanır. Bu katmanlar, resimdeki özellik haritasını çıkarmak için çeşitli filtreler uygulamaktadır.
- Aktivasyon Fonksiyonu: Her evrişimsel katmanın ardından ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu kullanılır. Bu, ağın daha derin öğrenmesine olanak tanır.
- Havuzlama Katmanları: VGG-19, boyut azaltmak için aynı VGG16'da olduğu gibi 2x2 boyutunda maksimum havuzlama *(max-pooling)* katmanları kullanır. Bu katmanlar, resim boyutunu azaltarak eğitim süresini düşürür ve aşırı uydurma *(overfitting)* riskini azaltmayı amaçlamaktadır.
- Tam Bağlantılı Katmanlar: VGG-19'un sonunda üç adet tam bağlantılı katman *(fully-connected layer)* bulunur. Bu katmanlar, sınıflandırma işlemi için kullanılır ve son çıktıyı veren katmanında da sınıflandırma yapacak nöronlar bulunur.



Şekil 7. VGG19 ağ mimarisi (Khattar, A., & Quadri, S. M. K., 2022)

1.6.3. ResNet50V2

ResNet50V2, ResNet (Residual Neural Network) mimarisi temel alınarak geliştirilmiş bir derin öğrenme mimarisidir. ResNet, Microsoft Research tarafından 2015 yılında tanıtılmış olup, ImageNet yarışmasında güzel başarılar elde eden bir derin öğrenme

mimarisidir. ResNet50V2, ResNet mimarisinin 50 katman içeren bir çeşididir ve V2 *(Version 2)* eklemesi, orijinal ResNet50'den türetilmiş başka bir sürüm olduğunu göstermektedir.

Asıl olarak, ResNet mimarisi derin sinir ağlarının eğitimi sırasında karşılaşılan kaybolan/gradyan patlaması gibi problemleri çözmeye yardımcı olan "bağlantılı atlamalar" veya "bağlantılı artıklar" *(skip connections, residual connections)* olarak adlandırılan özel bir mimariyi içermektedir. Bu mimarinin getirdiği avantaj ile daha derin ağlar eğitilebilir hale getirilir ve genellikle daha yüksek doğruluk elde edilmektedir.

Resnet50V2'nin temel mimarisini sıralayacak olursak:

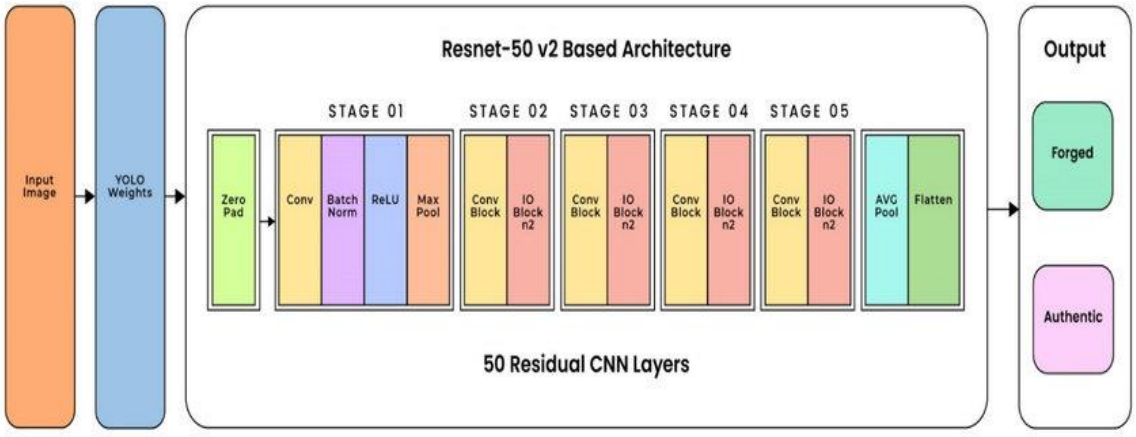
Katman Yapısı: Toplamda 50 katmandan oluşur. Bu katmanlar, evrişimsel katmanlar, toplama katmanları ve tam bağlantılı katmanları barındırmaktadır.

Bağlantılı Atlamalar: ResNet mimarisinde olduğu gibi, Resnet50V2'de de bağlantılı atlamalar *(skip connections)* kullanılmaktadır. Bu atlama bağlantıları, belirli katmanların çıktılarını daha sonraki katmanlardaki çıktılarla direk olarak birleştirerek, "artıkları" aktarmayı sağlar. Bu sayede, daha derin ağlar eğitilebilir hale gelmektedir.

Aktivasyon Fonksiyonu: ResNet mimarisinde olduğu gibi, Resnet50V2'de de ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu kullanılır.

Havuzlama Katmanları: Ağın mimarisinde, havuzlama katmanları *(pooling layers)* kullanılarak boyut düşürme işlemi gerçekleştirilmektedir. Bu sayede aşırı uydurmayı engellemek amaçlanmaktadır.

Tam Bağlantılı Katmanlar: Son evrişimsel katmandan sonra tam bağlantılı *(fully-connected)* katmanlar kullanılarak sınıflandırma işlemi sağlanmaktadır.

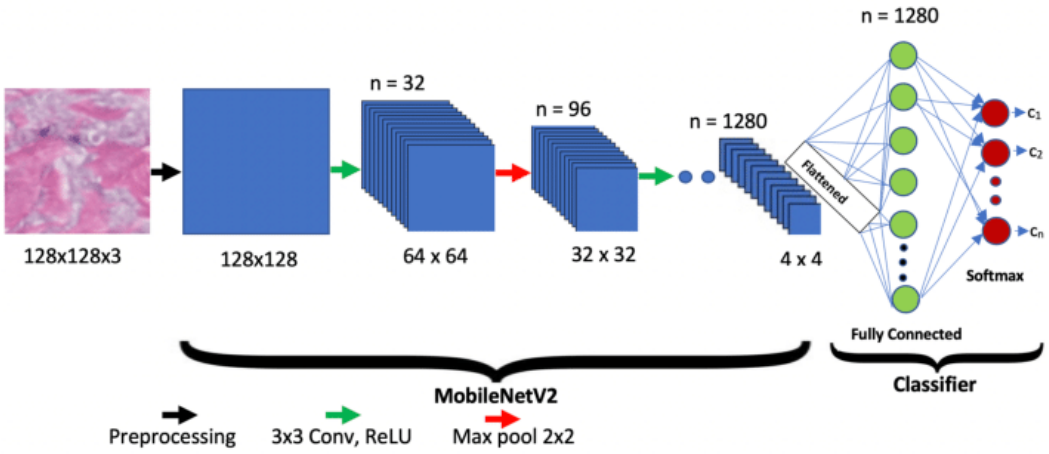


Şekil 8. ResNet50V2 ağ mimarisi (Qazi, E. U. H., Zia, T., & Almorjan, A., 2022)

1.6.4. MobileNetV2

MobileNetV2, Google tarafından üretilen ve mobil cihazlardaki düşük hesaplama potansiyeli ile yüksek performanslı görüntü sınıflandırma yapmayı amaçlayan bir derin öğrenme mimarisidir. MobileNetV2, özellikle akıllı telefonlar ve diğer kısıtlı kaynaklara sahip sistemlerde hızlı ve efektif nesne tespiti ve sınıflandırma yapmak için üretilmiştir. MobileNetV2'nin temel özellikleri şunlardır:

- Hafif Mimari: MobileNetV2, geleneksel derin sinir ağlarının karmaşıklığı yerine hafif ve basit bir derin öğrenme mimarisine sahiptir. Yüksek performanslı öğrenme yeteneği ile, daha az parametreye ve hesaplama gücüne ihtiyaç duymaktadır. Bu sayede öğrenme işlemini daha kısa sürelerde yapmayı amaçlamaktadır.
- Derinlemesine Ayrılabilir Evrişim *(Depthwise Separable Convolution)*: MobileNetV2, evrişim işlemlerini iki aşamalı bir şekilde gerçekleştiren Derinlemesine Ayrılabilir Evrişim adı verilen özel bir mimari kullanmaktadır. Bu mimari, geleneksel evrişimlerden daha az karmaşık ve hesaplama açısından daha iyi sonuçlar vermektedir.
- Aktivasyon Fonksiyonu: Modelde ReLU6 olarak adlandırılan sınırlı doğrusal bir aktivasyon fonksiyonu kullanılmaktadır. Bu, ağı belirli bir skalada tutarak tahminleri daha doğrusal bir hale getirir.
- Ters Artıklar *(Inverted Residuals)*: MobileNetV2, " Ters Artıklar " olarak bilinen bir mimari kullanır. Bu mimari, ağın daha derin ve karmaşık olmasını sağlamak için düşük boyutlu gömülü katmanları büyütmeyi ve efektif bir yöntemle öğrenmeyi hedeflemektedir.
- Havuzlama Katmanları: Modelde, birçok sayıda boyutu azaltmak ve öğrenim özelliklerini birleştirmek için havuzlama katmanı kullanılmaktadır.



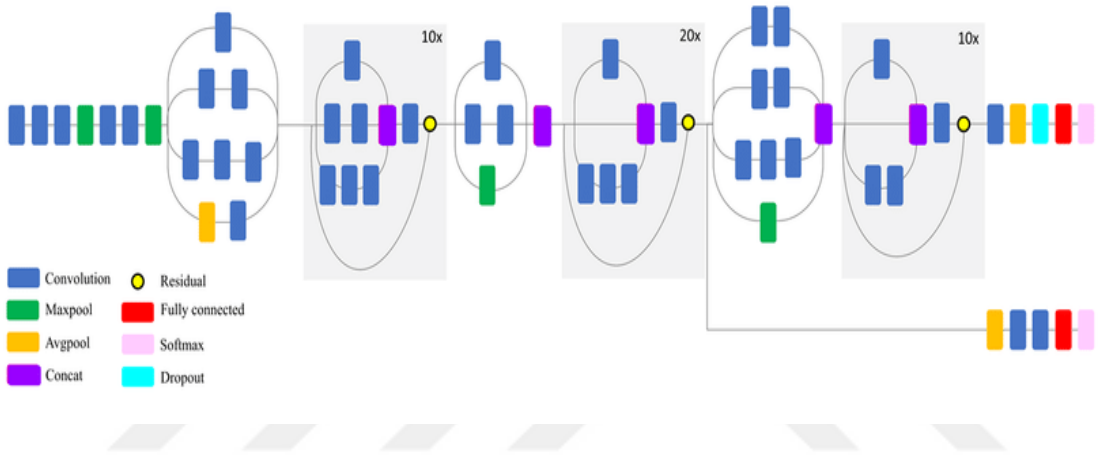
Şekil 9. MobileNetV2 ağ mimarisi (Akay, M., Du, Y., Sershen, C. L., Wu, M., Chen, T. Y., Assassi, S., ... & Akay, Y. M., 2021)

1.6.5. InceptionResNetV2

InceptionResNetV2, Google tarafından geliştirilen ve 2016 yılında tanıtılan bir derin öğrenme mimarisi olup bu mimari, Inception ve ResNet mimarilerinin birleşiminden oluşmaktadır. Birleşme sayesinde daha karmaşık bir yapıya sahip olan ağ ile derin öğrenme işlemi hedeflenmektedir. Başarıları ile bilinen iki mimariyi bir araya getirerek, daha derin ağlar elde etmek ve aynı zamanda hesaplama süresini optimuma indirmek hedeflenmiştir. InceptionResNetV2'nin temel özellikleri şunlardır:

- İki Başarılı Mimarinin Birleştirilmesi: Inception mimarisi, farklı boyutlarda paralel evrişimler yürüten "ince" filtrelerden oluşan derin bir mimari yapıya sahiptir ve bununla beraber daha efektif bir hesaplama imkânı sağlamaktadır. ResNet mimarisi ise bağlantılı atlamalar *(skip connections)* ile derin ağların eğitilmesini daha da kolaylaştıran bir mimariye sahiptir. InceptionResNetV2, bu iki yapıyı bir araya getirerek, her ikisinin getirilerinden faydalanmayı amaçlamaktadır.
- Bağlantılı Atlamalar: ResNet mimarisinde olduğu gibi, InceptionResNetV2'de de bağlantılı atlamalar *(skip connections)* kullanılır. Bu bağlantılar, belirli katmanların çıktılarını daha sonraki katmanlardaki çıktılarla doğrudan birleştirerek, "artıkları" aktarmayı sağlamaktadır. Bu sayede, daha derin ağlar eğitilebilir hale gelmektedir.

- Inception Blokları: Modelde, Inception mimarisi için özel olarak tasarlanan "Inception blokları" kullanılarak farklı boyutlarda paralel evrişimsel işlemleri gerçekleştirilmektedir.
- Aktivasyon Fonksiyonu: ReLU6 olarak adlandırılan sınırlı doğrusal bir aktivasyon fonksiyonu kullanılır.
- Havuzlama Katmanları: Ağın içerisinde, toplama katmanları *(pooling layers)* kullanılarak boyut azaltma işlemi gerçekleştirilir.



Şekil 10. InceptionResNetV2 ağ mimarisi (Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F., & Zhang, Y., 2018)

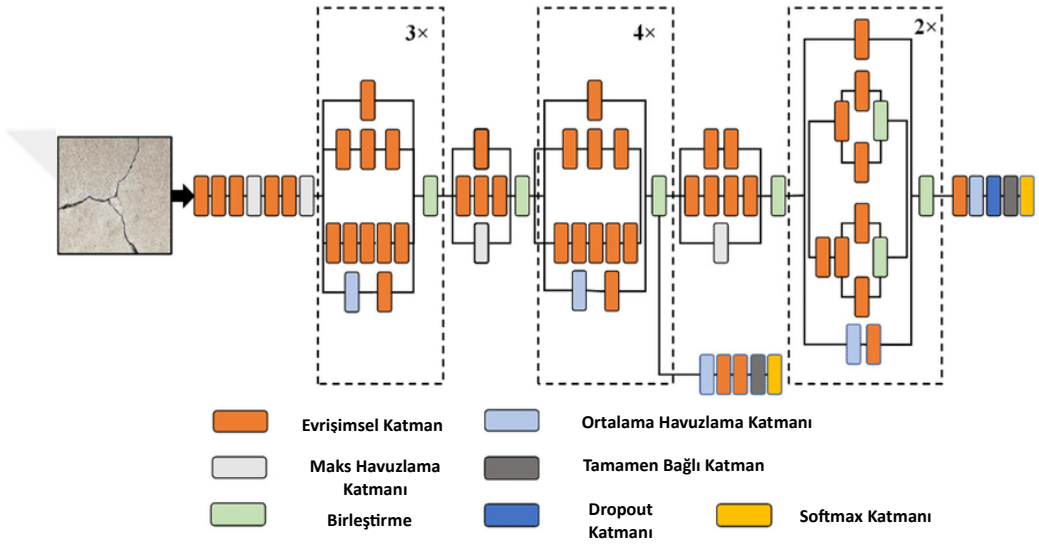
1.6.6. InceptionV3

InceptionV3, Google tarafından geliştirilen ve 2015 yılında ortaya çıkarılan derin öğrenme mimarisidir. Bu mimari daha önce üretilen Inception mimarilerinden esinlenerek üretilmiştir. InceptionV3, sınıflandırma yaklaşımları ve görsel tanıma için ImageNet veri kümesi gibi büyük veri kümesi üzerinde yüksek doğruluk elde etmek hedeflenmiştir.

InceptionV3'ün mimari yapısı şu şekildedir:

- İnce ve Derin Filtreler: Inception mimarisi, farklı boyutlarda paralel evrişimsel katmanları birleştiren bir yapıya sahiptir. InceptionV3, ince (1x1 boyutunda) ve derin (3x3 ve 5x5 boyutlarında) filtreleri birleştirerek verimli hesaplama sağlar ve daha karmaşık özellikleri öğrenmeye imkân tanır.
- Havuzlama Katmanları: InceptionV3'te, boyut azaltma ve özellik haritalarını birleştirme işlemleri için birçok toplama katmanları kullanılmaktadır.

- Aktivasyon Fonksiyonu: Modelde ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu kullanılmaktadır.
- Küçük Boyutlu Evrişim katmanları: InceptionV3, 1x1 boyutlu evrişim katmanlarını kullanarak boyut azaltma ve öğrenme süresini optimuma indirmeyi hedefler.
- Bağlantılı Atlamalar: InceptionV3'te, ResNet mimarisinde olduğu gibi bağlantılı atlamalar *(skip connections)* kullanılmaz. Bunun yerine, InceptionV3, düzgün bir derinlik ve genişlik boyutuna sahip homojen bir yapı kullanır.



Şekil 11. InceptionV3 ağ mimarisi (Ali, L., Alnajjar, F., Jassmi, H. A., Gocho, M., Khan, W., & Serhani, M. A., 2021)

1.6.7. ResNet50

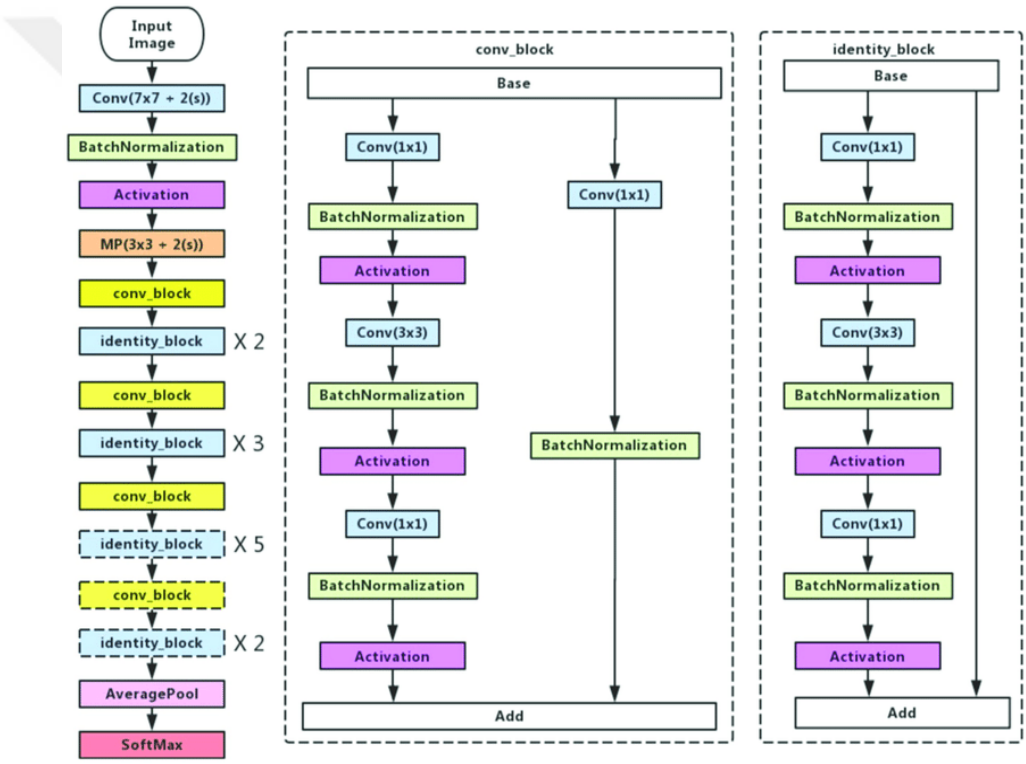
ResNet50, Microsoft tarafından 2015 yılında tanıtılan yapay zekâ mimarisidir. Adı "Residual Neural Network" ifadesinden gelen bu mimari, derin ağların eğitimi esnasında ortaya çıkan kaybolan veya patlayan gradyan problemlerini çözmeyi amaçlamaktadır. Bundan dolayı daha derin ağlar eğitilebilir hale gelmektedir ve daha yüksek doğruluk elde etme imkânı sağlanmaktadır.

ResNet50'nin genel mimarisi şu şekildedir:

- Bağlantılı Atlamalar *(Skip Connections)*: ResNet mimarisi, bağlantılı atlamalar *(skip connections)* olarak adlandırılan özel bir yapıyı kullanır. Bu atlama bağlantıları, belirli katmanların çıktılarını daha sonraki katmanlardaki çıktılarla

direk olarak birleştirerek, "artıkları" aktarmayı sağlar. Bu sayede, daha derin ağlar eğitilebilir hale gelmektedir. Bu yapı, daha derin ağların eğitilebilir hale gelmesini ve gradyanların sorunsuz bir şekilde geriye yayılmasını sağlamaktadır.

- İnce Evrişimsel Katmanlar: ResNet50, ince (1x1 boyutunda) evrişimsel katmanları kullanarak boyut azaltma ve öğrenmeyi optimize etmektedir.
- Aktivasyon Fonksiyonu: Modelde ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu tercih edilmektedir.
- Havuzlama Katmanları: ResNet50, boyut azaltma ve özellik haritalarını birleştirme işlemleri için çeşitli toplama katmanları kullanır.



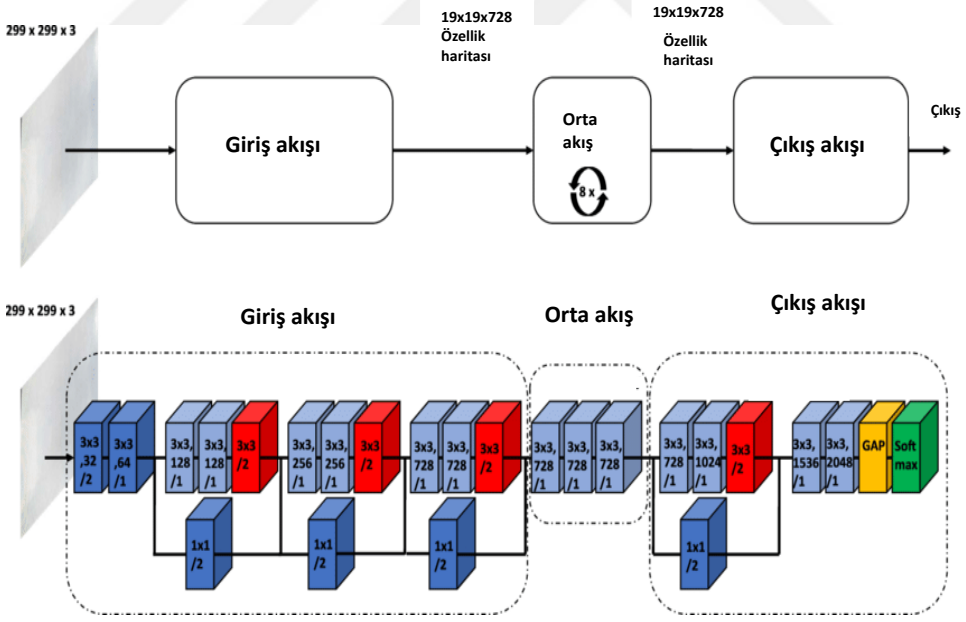
Şekil 12. ResNet50 ağ mimarisi (Ji, Q., Huang, J., He, W., & Sun, Y.,2019)

1.6.8. Xception

Xception, Google tarafından geliştirilen bir derin öğrenme mimarisidir. "Extreme Inception" anlamına gelen bu mimari Inception mimarisi göz önüne alınarak tasarlanmıştır. Inception mimarisindem farklı bir yaklaşım uygulanarak efektif olmayı amaçlayan bir mimari tasarlanmıştır.

Xception'ın genel yapı taşları şu şekildedir:

- Dışbükey Evrişim Katmanları: Xception, Inception mimarisindeki 3x3 ve 5x5 boyutundaki evrişim katmanlarının yerine, 1x1 boyutundaki dışbükey evrişim katmanları kullanmaktadır. Dışbükey evrişim katmanları, daha kullanışlı bir şekilde özellik çıkarımı sağlamaktadır. Bununla beraber daha az parametreye ihtiyaç duyan mimari, eğitim süresini kısaltmayı amaçlamaktadır.
- Kanal Dışı Bağlantılı Atlamalar: Xception, Inception mimarisindeki paralel kanal içi bağlantıların yerine, kanal dışı bağlantılı atlamalar *(depthwise separable convolutions)* kullanır. Bu bağlantılar sayesinde hesaplama açısından daha efektif bir yapıya sahip olması ve ağın daha hızlı çalışması hedeflenmektedir.
- Aktivasyon Fonksiyonu: Modelde ReLU *(Rectified Linear Unit)* aktivasyon fonksiyonu tercih edilmiştir.
- Havuzlama Katmanları: Xception, boyut azaltma ve özellik haritalarını birleştirme işlemleri için çeşitli toplama katmanları kullanır.



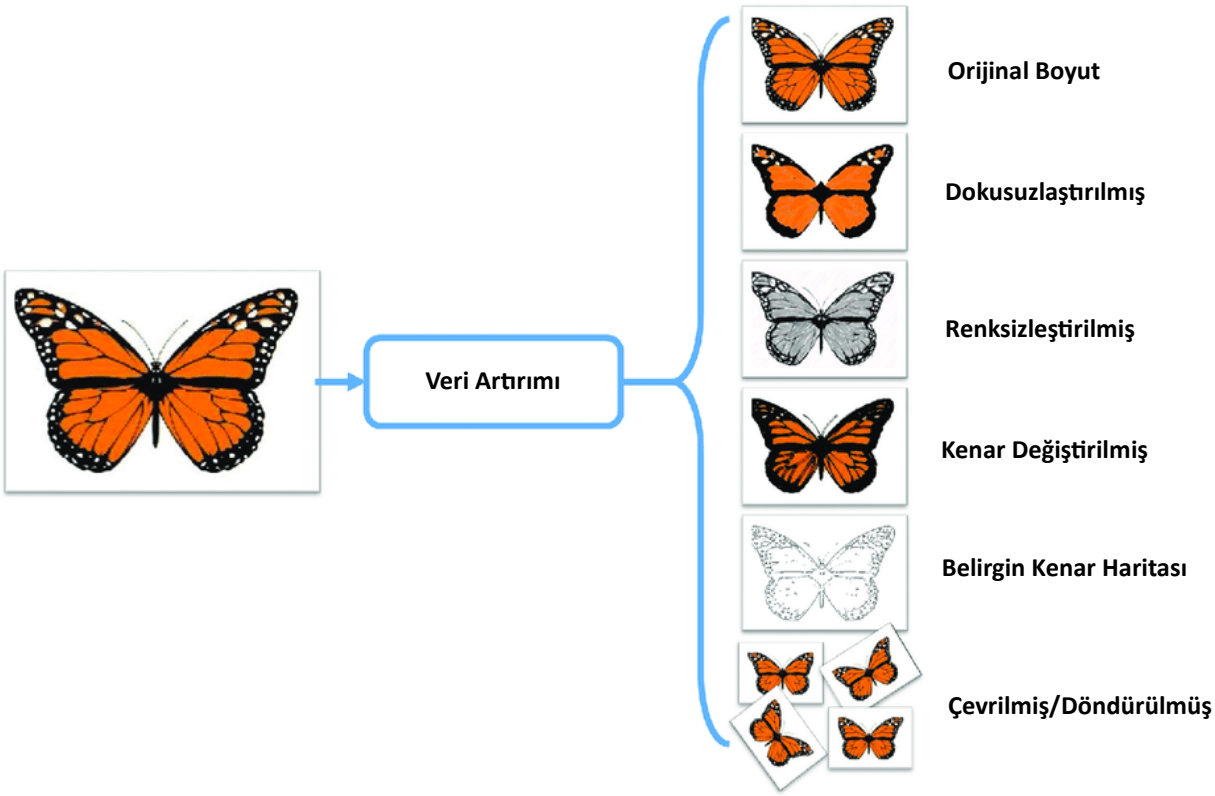
Şekil 13. Xception ağ mimarisi (Westphal, E., & Seitz, H., 2021)

1.7. Veri Artırımı *(Data Augmentation)*

Derin öğrenme tekniği, eğitim işlemi uygulanırken büyük veri kümelerin ve karmaşık yapıların kullanılmasını gerektiren bir makine öğrenme işlemidir. Bu sebeple veri setinin

büyüklüğü ve karmaşıklığı derin öğrenme işlemi uygulanırken önemli bir rol oynamaktadır. Veri artırımı tekniği, tıp alanında olduğu gibi veri erişiminin problemli olduğu bilim dallarında sınırlı veri problemine çözüm sunmaktadır (Shorten, C., & Khoshgoftaar, T. M., 2019). Veri artırımının mantığı eldeki veriyi kullanılan bilgisayar fonksiyonları yardımı ile ana özellik haritasından çok farklılaşmayacak şekilde değiştirip çoğaltmaktır. Bu sayede eldeki az sayıda olan veriyle daha iyi performans sunan modeller eğitilebilmektedir. Veri artırımı yönteminin neden önemli olduğunu birkaç madde ile özetleyecek olursak:

- Genelleme performansında iyileşme: Farklı tip veriler ile eğitilen mimari, farklı ve yeni tip veriler ile daha iyi sonuçlar verebilmektedir. Veri artırımı tekniği, modelin daha önce görmediği örnekleri daha sonra gördüğünde doğru bir şekilde sınıflama yapmasına ve tahminde bulunmasına olanak sağlar.
- Aşırı uydurmayı *(overfitting)* azaltma: Modelin eğitim modellerine aşırı uyum sağlaması derin öğrenme algoritmalarında istenen bir durum değildir. Bu durum modelin gerçek veri karşısında genelleme yapabilme kabiliyetini düşürmektedir. Veri aktarımı metodu modelin bu gibi durumlardan kaçınmasını sağlamaktadır. Veri alanında büyütme sağlamanın performansı artırmak ve fazla uydurmayı azaltmak için büyük fayda sağladığı tespit edilmiştir (Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D., 2016).
- Dirençli Modeller: Veri artırımı tekniği ile modelin eğitim verilerindeki küçük değişikliklere karşı direnci artar. Bununla beraber model, aynı özellik haritalarını içeren farklı verilerde daha başarılı sonuçlar gösterme eğilimindedir.
- Veri Azlığı: Derin öğrenme algoritmaları uygulanırken eğitim veri setinin büyük olması modelin performansı açısından çok önemlidir. Eğer bir model oluşturulmak isteniyorsa ve eğitim verisi az ise model istenilen başarıyı sağlamayabilir. Buna çözüm olarak veri artırımı yöntemi iyi bir alternatif sunar. Eldeki verileri çoğaltarak daha efektif bir şekilde kullanma imkânı sağlamaktadır.
- Nadir Sınıflar: Derin öğrenme algoritmaları eğitilirken eğitim verisindeki bazı sınıfların verileri yeteri kadar olmayabilir. Bu gibi durumlarda eğitilen model nadir sınıfların özellik haritalarına aşina olamayacağı için performansı düşecektir. Veri artırımı tekniği ile bu gibi az veriye sahip sınıflar çoğaltılarak modelin daha dengeli sonuçlar üretmesi sağlanabilmektedir.



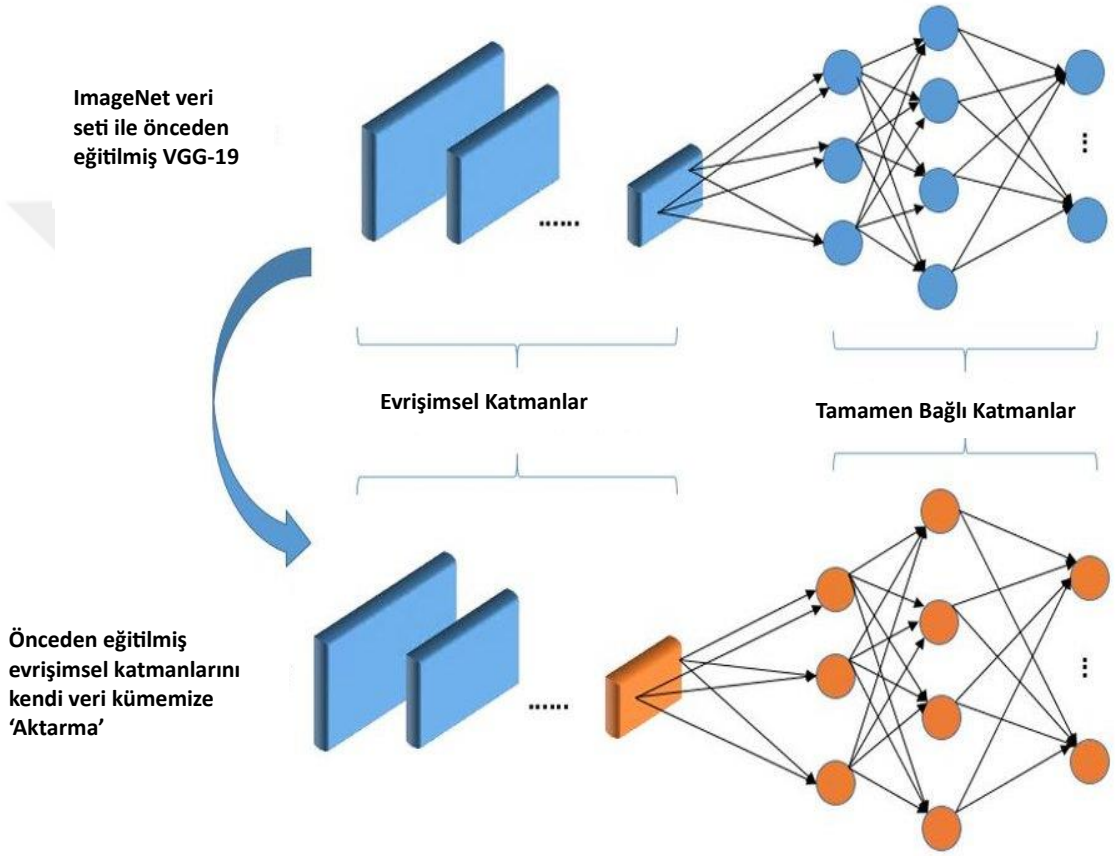
Şekil 14. Görüntüde veri artırımı (Ahmad, J., Muhammad, K., & Baik, S. W., 2017)

Veri artırımı teknikler birçok alanda uygulanabilmektedir. Özellikle görüntü, ses ve metin verilerinin artırımı alanında başarılı sonuçlar vermektedir.

1.8. Öğrenim Aktarımı *(Transfer Learning)*

Öğrenim aktarımının amacı, kaynak modeldeki bilgiden yararlanılarak öğrenim durumunu daha da fazla geliştirmektir (Torrey, L., & Shavlik, J., 2010). Makine öğrenimi uygulamalarında istenilen durum kullanılan girdi verilerinin ve sonradan kullanılacak gerçek verilerin benzer olmasıdır (Zhu vd., 2005). Geniş uygulama imkanları sayesinde transfer öğrenme uygulaması, makine öğreniminde popüler ve önü açık bir yöntem olarak düşünülebilmektedir (Weiss vd., 2016). Bu sebepten dolayı eğitilen modelin gerçek veriler ile daha iyi sonuç verebilmesi için öğrenim aktarımı tekniği uygulanmaktadır. Bu yöntem ile modelin eğitim için ihtiyaç duyduğu büyük çaplı bir eğitim verisine gerek kalmaz ve eldeki eğitim verileri de etkili bir şekilde kullanılmış olur (Han, D., Liu, Q., & Fan, W., 2018). Teknik uygulanırken eldeki verinin çoğaltılıp tekrar modele uygulanması da yine olumlu bir

yaklaşımdır. Transfer öğrenme yönteminde insanların yeni sorunlara çözüm yolu ararken önceden edindiği deneyimleri kullanmasından ilham alınmıştır (Lu vd., 2015). Ayrıca başka bir uygulama için çalışılmış veri kümelerinin özelliklerinin, mevcut çalışılan modele aktarılması bu yöntem ile mümkündür. Önceden öğrenim işlemi görmüş ağın ağırlıklarından faydalanılarak tekrar eğitim görmesi ağın performansı açısından olumlu yansımıştır (Yosinski, J., Clune, J., Bengio, Y. ve Lipson, H., 2014).

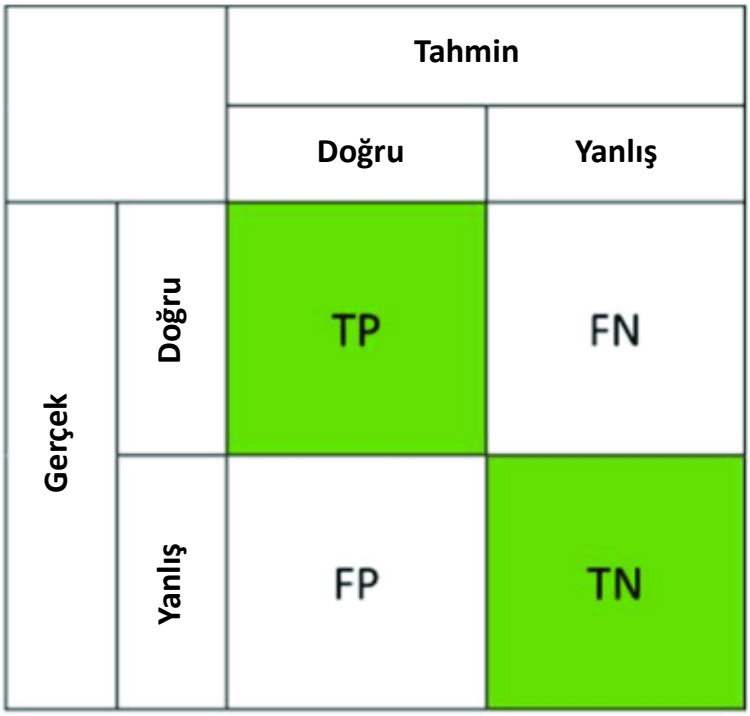


Şekil 15. Öğrenim aktarımı örneklemesi (Wang, Y., Lucas, M., Furst, J., Fawzi, A. A., & Raicu, D., 2020)

Öğrenim aktarımı yöntemini özetleyecek olursak, eğitim setiyle eğitilmiş ve ağırlıkları belirlenmiş bir modelin farklı veya gerçek verilerde daha iyi sonuçlar verebilmesi için mevcut veriler çoğaltılarak veya aynı özellik haritasına sahip farklı veriler tekrar modele öğretilerek daha başarılı sonuçlar almayı amaçlamaktır. Bu sayede girdi verisi büyük boyutlu olmasa bile daha etkili sonuçlar almak mümkündür.

1.9. Sonuç Değerlendirme Metrikleri

Derin öğrenme modelinin başarısı yorumlanırken eğitim sonunda yapılan test verilerine göre verdiği doğru ve yanlış cevaplar bir matris haline getirilir. Bu matris literatürde hata matrisi *(confusion matrix)* olarak bilinmektedir. Hata matrisi, ilgili problemdeki sınıf sayısı boyutundadır.



Şekil 16. Hata matrisi (Le, T., Lee, M. Y., Park, J. R., & Baik, S. W., 2018)

Modelin başarısı köşegenlerde oluşan yığılmaya göre yorumlanabilmektedir. Hata matrisinin sonuçlarına göre genel doğruluk *(overall accuracy)*, duyarlılık *(recall)*, kesinlik *(precission)*, F1-skoru gibi modelin başarısı ve eğilimi hakkında yorum yapabilecek parametreler üretilebilmektedir.

Genel doğruluk skoru, modelin verdiği doğru cevapların bütün verilen cevaplara bölünmesi ile elde edilmektedir. Matrise göre şöyledir:

$$G.D.S. = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \quad (5)$$

Duyarlılık skoru ise modelin doğru olarak tahmin etmesi gereken cevapların ne kadarını doğru olarak tahmin ettiğini gösterir. Matrise göre şöyledir:

$$Duyarlılık = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6}$$

Kesinlik skoru modelin doğru olarak tahmin ettiği cevapların gerçekte ne kadarının doğru olduğunu gösteren bir metriktir. Matrise göre şöyledir:

$$Kesinlik = \frac{\mathrm{TP}}{TP+FP} \tag{7}$$

F1-skoru, kesinlik metriği ile duyarlılık metriğinin harmonik ortalamasını göstermektedir. Matrise göre şöyledir:

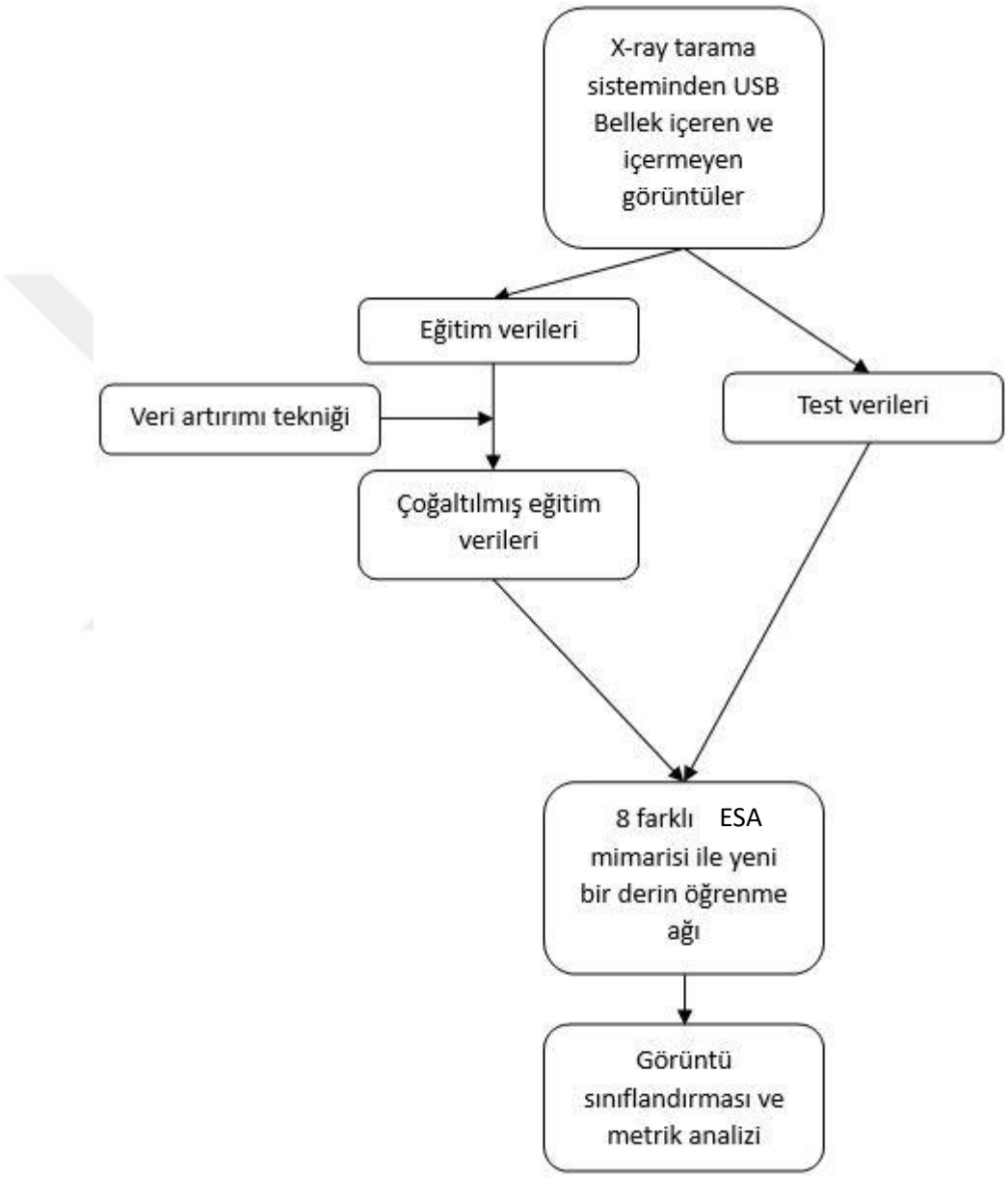
$$F1 - Skoru = \frac{2*(\mathrm{Duyarlılık}*\mathrm{Kesinlik})}{(Duyarlılık+Kesinlik} \tag{8}$$

2. YAPILAN ÇALIŞMALAR

Çalışmada, X-ray bagaj tarama sistemlerinden elde edilen 1217 adet görüntünün USB bellek içerip içermediği sınıflandırmasını yapmak için 8 farklı ESA mimarisi kullanılmıştır. 8 farklı ESA mimarisi ile oluşturulan yeni ağ önce normal verilerle eğitim-test işlemi görüp oluşan skorlar not edilmiştir. Daha sonra aynı şekilde yeniden oluşturulan 8 adet model artırılmış veriler kullanılarak eğitim-test işlemi görmüştür. Elde edilen skorlar karşılaştırılmak üzere not edilmiştir. Son olarak normal veriler ile eğitilen ağırlıkları belirlenmiş olan 8 adet model artırılmış veriler ile eğitim-test işlemi görüp öğrenim aktarımı metodu uygulanmıştır. Elde edilen sonuçlar not edilmiştir. Toplamda 3 farklı metot ile yaklaşım yapılmış ve her modelin oluşturduğu skorlar ayrıntılı bir şekilde incelenmiştir.

Şekil 17. Tip-1 çalışma yöntemi

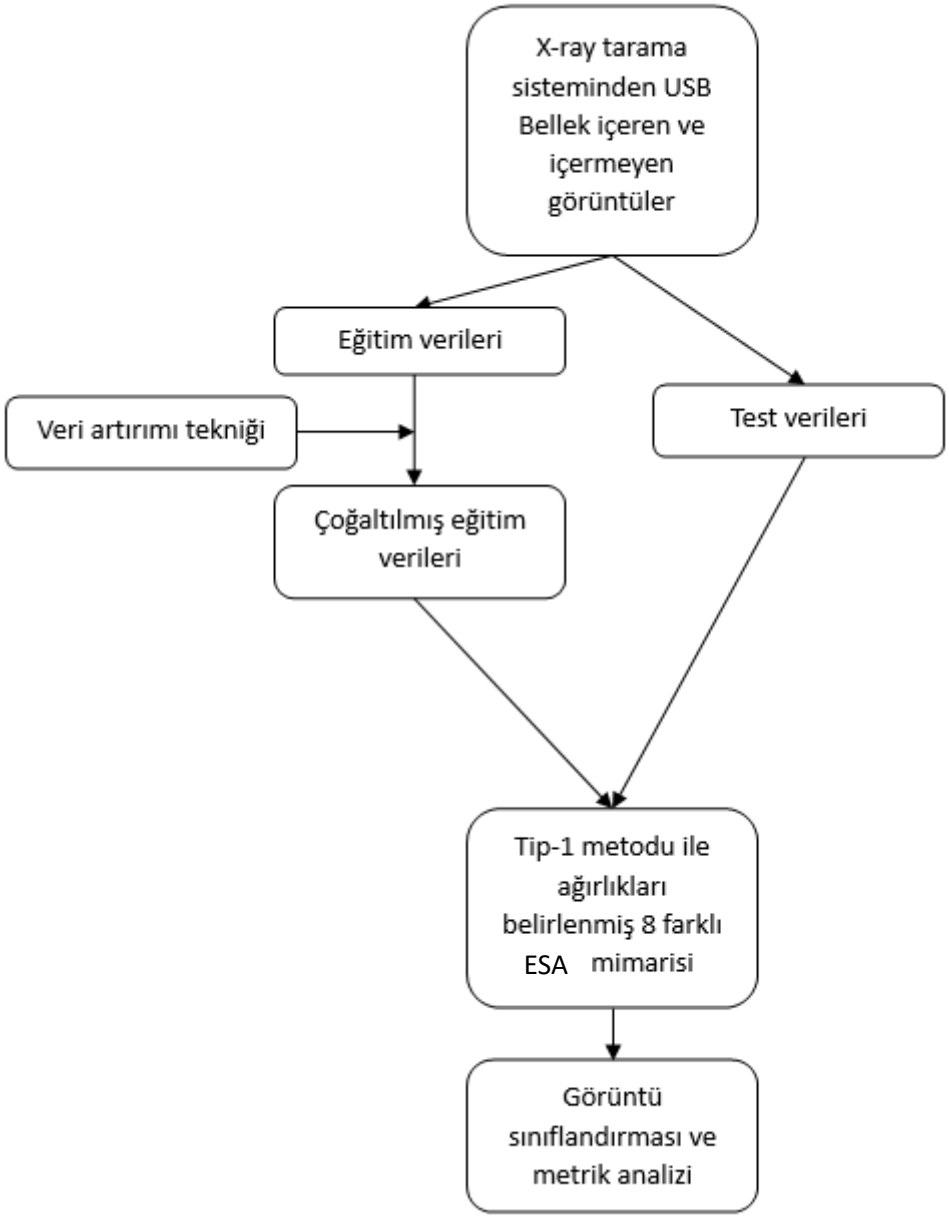
Tip-1 yönteminde önce X-ray tarama cihazından elde edilen görüntüler sınıfsal olarak ayrılıp daha sonrasında %85-15 oranında eğitim ve test verisi olacak şekilde düzenlenmiştir. Bu düzenlenen veriler ana hattını ESA mimarilerinin oluşturduğu modellere 224x224x3 boyutunda olacak şekilde öğretilmiştir. Elde edilen çıktı verilerinin metrikleri incelenmiştir.



Şekil 18. Tip-2 çalışma yöntemi

Tip-2 yönteminde X-ray tarama cihazından elde edilen görüntüler sınıfsal olarak ayrılıp daha sonrasında %85-15 oranında eğitim ve test verisi olacak şekilde düzenlenmiştir.

Düzenlenen eğitim veri kümeleri veri artırımı yöntemi ile çoğaltılmıştır. Çoğaltılan yeni veriler ve normal şekilde oluşturulan test verileri ana hattını ESA mimarilerinin oluşturduğu modellere 224x224x3 boyutunda olacak şekilde öğretilmiştir. Elde edilen çıktı verilerinin metrikleri incelenmiştir. Bu teknik ile veri artırımı tekniğinin derin öğrenme mimarisinin ürettiği sonuçları ne derecede etkilediği gözlenmiştir.

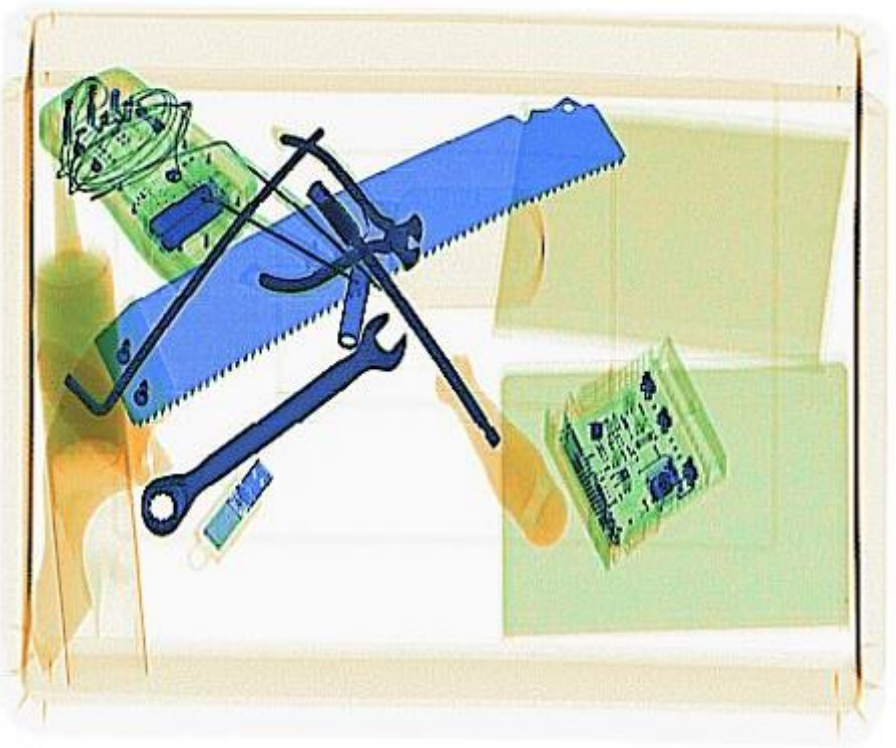


Şekil 19. Tip-3 çalışma yöntemi

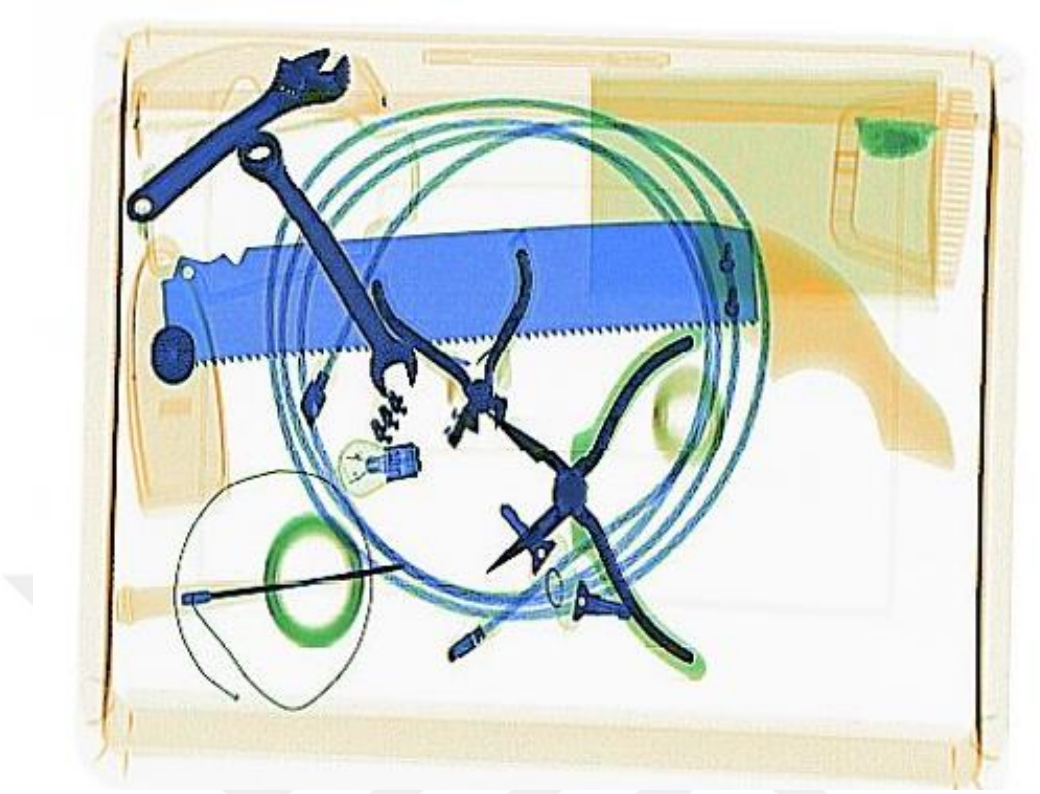
Tip-3 yönteminde X-ray tarama cihazından elde edilen görüntüler sınıfsal olarak ayrılıp daha sonrasında %85-15 oranında eğitim ve test verisi olacak şekilde düzenlenmiştir. Düzenlenen eğitim veri kümeleri veri artırımı yöntemi ile çoğaltılmıştır. Çoğaltılan yeni veriler ve Tip-1 metodu ile ağırlıkları belirlenmiş 8 farklı ESA mimarisine 224x224x3 boyutunda olacak şekilde öğretilmiştir. Elde edilen çıktı verilerinin metrikleri incelenmiştir. Bu yöntem ile öğrenim aktarımı tekniğinin model ve sınıflandırma problemi üzerindeki etkileri araştırılmıştır.

2.1. Verilerin Hazırlanması

Görüntüler Rapiscan 620DV markalı X-ray bagaj tarama cihazıyla oluşturulmuştur. Görüntüler oluşturulurken USB Bellek ile çeşitli türde materyal de kullanılmıştır. Her taramada rastgele koyulan materyaller ve USB Bellekler ile homojen bir görüntü kümesi elde edilmeye çalışılmıştır. USB Bellek içeren görüntü kümesine bir veya birden fazla USB Bellek koyulup X-ray cihazından RGB görüntüler elde edilmiştir. USB Bellek içermeyen görüntülere ise yine USB Bellek içeren görüntülerdeki materyaller rastgele koyulmuştur.



Şekil 20. USB Bellek içeren bir görüntü



Şekil 21. USB Bellek içermeyen bir görüntü

Görüntüler X-ray tarama sisteminden elde edildikten sonra eğitim ve test olmak üzere ikiye ayrılmıştır. Eğitim için olan veri kümesinde 670 USB Bellek içeren görüntü ve 333 adet USB Bellek içermeyen görüntü bulunmaktadır. Test için ayrılan veri kümesinde ise 137 adet USB Bellek içeren görüntü, 77 adet USB Bellek içermeyen görüntü bulunmaktadır.

2.2. Çalışılan Ortam

Tez çalışması derin öğrenme algoritması olduğu için yüksek işlem gücüne ihtiyaç duymaktadır. Buna bağlı olarak daha hızlı çalışılabilmesi için derin öğrenme algoritmasında grafik işlem birimi (GPU) olan NVIDIA GeForce GTX 1650 SUPER ekran kartı kullanılmıştır. Tez çalışması Python programlama dili ile oluşturulmuştur. Anaconda adlı programın bir ürünü olan Jupyter Notebook programı yazılan Python kodlarını çalıştırmak için kullanılmıştır. Tezde os, pandas, matplotlib, tensorflow, numpy, sklearn kütüphanelerinden faydalanılmıştır. Tensorflow'un bir kütüphanesi olan keras kütühanesi de model oluşturulurken ve model parametreleri belirlenirken tezde yoğun olarak kullanılmıştır.

2.3. Veri Kümelerinin Çalışmaya Hazır Hale Getirilmesi

Elde edilen görüntüler eğitim ve test verileri şeklinde ayrıldıktan sonra modele öğretilmeden önce hazır hale getirilmiştir. Veri setleri kullanılmak üzere normal ve veri artırımı tekniği ile çoğaltılmış olacak şekilde iki tip olarak hazırlanmıştır.

2.3.1. Normal Veri Setlerinin Oluşturulması

Normal veri setinin oluşturulması için generate_data adlı bir fonksiyon oluşturulmuştur. Fonksiyon, ilgili dosya yollarından görüntü kümelerini alıp eğitim ve test verisi olarak hazır hale getirmeyi hedeflemektedir.

```
In [2]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
        import cv2
        from PIL import ImageFilter

        height, width = 224, 224
        batch_size=64


        def generate_data(DIR):
            datagen = ImageDataGenerator(rescale=1./255.)

            generator = datagen.flow_from_directory(
                DIR,
                batch_size=batch_size,
                shuffle=True,
                seed=42,
                class_mode='binary',
                target_size=(height, width),
                classes={'flashsiz': 0, 'flashli': 1}
            )
            return generator

        TRAINING_DIR = 'Desktop/v1_training'
        TESTING_DIR = 'Desktop/v1_test'

        train_generator = generate_data(TRAINING_DIR)
        test_generator = generate_data(TESTING_DIR)

        total_image = np.concatenate([train_generator.labels,test_generator.labels])

        print('\n\n',{'flashsiz':len(np.where(total_image==0)[0]),
              'flashli':len(np.where(total_image==1)[0])})

        Found 1003 images belonging to 2 classes.
        Found 214 images belonging to 2 classes.


         {'flashsiz': 410, 'flashli': 807}
```

Şekil 22. Normal eğitim ve test veri kümelerini oluşturan generate_data fonksiyonu

Generate_data fonksiyonu ilgili dosya yolundan aldığı veriyi yeniden boyutlandırma işlemi ile boyutlandırıp daha sonra modele girmeden önce verinin yığın sayısını, karıştırma işlemini, sınıflarını düzenlemektedir. Bu sayede normal eğitim ve test verileri modele eğitime hazır hale gelmektedir.

2.3.2. Veri Artırımı Yöntemi İle Eğitim Veri Setinin Oluşturulması

Bir önceki adımda oluşturulan test verisi aynı kalırken eğitim verisi veri artırımı yöntemi ile tekrar oluşturulmuştur. Bu yöntem generate_data_augmented adlı oluşturulan fonksiyon ile sağlanmıştır.

```
In [3]: def generate_data_augmented(DIR):
            datagen = ImageDataGenerator(
                rescale=1./255.,
                zoom_range=0.1,
                rotation_range=20,
                width_shift_range=0.1,
                height_shift_range=0.1,
                horizontal_flip = True
            )
            generator = datagen.flow_from_directory(
                TRAINING_DIR,
                batch_size=batch_size,
                seed=42,
                class_mode='binary',
                target_size=(height, width),
                classes={'flashsiz': 0, 'flashli': 1}
            )
            return generator

        aug_train_generator = generate_data_augmented(TRAINING_DIR)

        Found 1003 images belonging to 2 classes.
```

Şekil 23. Veri arttırımı tekniği için hazırlanan generate_data_augmented fonksiyonu

Fonksiyonda veri arttırımı tekniği uygulanacak olan veri, bazı parametreler yardımıyla düzenlenmiştir. Bu fonksiyon yardımıyla yeni oluşturulan verideki görüntülerin bir kısmı yakınlaşma, döndürme, dikey eksende döndürülme, yatay eksende kayma uygulaması, dikey eksende kayma uygulaması şeklinde eğitime hazır hale getirilmiştir.

2.4. Modelin Oluşturulması

Çalışmada 8 farklı ESA modeli oluşturulmuştur. Bu modeller veri artırımı tekniği ile eğitim için ayrıca yeniden oluşturulmuştur. Öğrenim aktarımı tekniği için normal veriler ile eğitilmiş Tip – 1 modeli tekrar veri arttırımı tekniği ile çoğaltılmış veriler ile eğitilmiştir.

2.4.1. VGG16 Temelli Modelin Oluşturulması

VGG16 ile oluşturulan modelin evrişimsel katmanı başlangıç ağırlıkları 'imagenet' olan VGG16 mimarisi ile oluşturulmuştur. Evrişim katmanından sonra düzleştirme işlemi için 'Flatten' katmanı eklenmiştir. Daha sonra 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı olacak şekilde modele entegre edilmiştir. Son olarak modelin çıkış katmanı 2 adet 'softmax' aktivasyon fonksiyonu ile oluşturulmuştur. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır. Metrik ayarları 'accuracy' doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu 'Adam' olarak seçilmiştir.

```
In [5]: from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.vgg16.VGG16(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_vgg16 = tf.keras.Sequential()
model_vgg16.add(base_model)
model_vgg16.add(tf.keras.layers.GlobalAveragePooling2D())

model_vgg16.add(tf.keras.layers.Flatten())
model_vgg16.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg16.add(tf.keras.layers.Dropout(0.5))
model_vgg16.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg16.add(tf.keras.layers.Dropout(0.5))

model_vgg16.add(tf.keras.layers.Dense(2, activation='softmax'))

model_vgg16.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_vgg16.summary()
```

Şekil 24. VGG16 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 14,912,322
Trainable params: 197,634
Non-trainable params: 14,714,688
_________________________________________________________________
```

Şekil 25. VGG16 temelli oluşturulan ağ mimarisi outputu

Şekil 25'te görüldüğü gibi girilen hiperparametre değerlerine göre modelde 14912322 adet parametre oluşmuştur.

2.4.2. VGG19 Temelli Modelin Oluşturulması

VGG19 temelli modeli oluştururken evrişimsel katmana VGG19'un mimarisi yerleştirilmiştir. Aynı VGG16'da olduğu gibi Evrişim katmanından sonra düzleştirme işlemi için 'Flatten' katmanı eklenmiştir. Daha sonra 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele entegre edilmiştir. Son olarak modelin çıkış katmanı 2 adet 'softmax' aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır. Metrik ayarları 'accuracy' doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu 'Adam' olarak seçilmiştir.

```
from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = VGG19(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_vgg19 = tf.keras.Sequential()
model_vgg19.add(base_model)
model_vgg19.add(tf.keras.layers.GlobalAveragePooling2D())

model_vgg19.add(tf.keras.layers.Flatten())
model_vgg19.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg19.add(tf.keras.layers.Dropout(0.5))
model_vgg19.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg19.add(tf.keras.layers.Dropout(0.5))

model_vgg19.add(tf.keras.layers.Dense(2, activation='softmax'))

model_vgg19.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_vgg19.summary()
```

Şekil 26. VGG19 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 20,222,018
Trainable params: 197,634
Non-trainable params: 20,024,384
_________________________________________________________________
```

Şekil 27. VGG19 temelli oluşturulan ağ mimarisi outputu

Girilen hiperparametre değerlerine göre modelde 20222018 adet parametre oluştuğu görülmüştür.

2.4.3. ResNet50V2 Temelli Modelin Oluşturulması

ResNet *(Residual Neural Network)* mimarisi temel alınarak geliştirilmiş ResNet50V2 mimarisi evrişimsel katman için ana kısım olacak şekilde oluşturulmuştur. Daha önce oluşturulan modellerde olduğu gibi Evrişim katmanından sonra düzleştirme yöntemi için 'Flatten' katmanı modele eklenmiştir. Daha sonra 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele entegre edilmiştir. Son olarak modelin çıkış katmanı 2 adet 'softmax' aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır. Metrik ayarları 'accuracy' doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu 'Adam' olarak seçilmiştir.

```
from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.ResNet50V2(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_ResNet50V2 = tf.keras.Sequential()
model_ResNet50V2.add(base_model)
model_ResNet50V2.add(tf.keras.layers.GlobalAveragePooling2D())

model_ResNet50V2.add(tf.keras.layers.Flatten())
model_ResNet50V2.add(tf.keras.layers.Dense(256, activation='relu'))
model_ResNet50V2.add(tf.keras.layers.Dropout(0.5))
model_ResNet50V2.add(tf.keras.layers.Dense(256, activation='relu'))
model_ResNet50V2.add(tf.keras.layers.Dropout(0.5))

model_ResNet50V2.add(tf.keras.layers.Dense(2, activation='softmax'))

model_ResNet50V2.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_ResNet50V2.summary()
```

Şekil 28. ResNet50V2 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 24,155,650
Trainable params: 590,850
Non-trainable params: 23,564,800
_________________________________________________________________
```

Şekil 29. ResNet50V2 temelli oluşturulan ağ mimarisi outputu

Girilen hiperparametre değerlerine göre modelde 24155650 adet parametre oluşmuştur.

2.4.4. MobileNetV2 Temelli Modelin Oluşturulması

Bu modelin evrişim katmanını, Google tarafından üretilen ve mobil cihazlardaki düşük hesaplama potansiyeli ile yüksek performanslı görüntü sınıflandırma yapmayı amaçlayan MobileNetV2 oluşturmuştur. Önceki modellerde oluşturulduğu gibi Evrişim katmanından sonra düzleştirme yöntemi için 'Flatten' katmanı modele eklenmiştir. Sonrasında 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele eklenmiştir. Modelin son katmanı ise 2 adet 'softmax' aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır.

Metrik ayarları ‘accuracy’ doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu ‘Adam’ olarak seçilmiştir.

```
In [5]: from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_MobileNetV2 = tf.keras.Sequential()
model_MobileNetV2.add(base_model)
model_MobileNetV2.add(tf.keras.layers.GlobalAveragePooling2D())

model_MobileNetV2.add(tf.keras.layers.Flatten())
model_MobileNetV2.add(tf.keras.layers.Dense(256, activation='relu'))
model_MobileNetV2.add(tf.keras.layers.Dropout(0.5))
model_MobileNetV2.add(tf.keras.layers.Dense(256, activation='relu'))
model_MobileNetV2.add(tf.keras.layers.Dropout(0.5))

model_MobileNetV2.add(tf.keras.layers.Dense(2, activation='softmax'))

model_MobileNetV2.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_MobileNetV2.summary()
```

Şekil 30. MobileNetV2 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 7, 7, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 1280)              0

 dense (Dense)               (None, 256)               327936

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 2,652,226
Trainable params: 394,242
Non-trainable params: 2,257,984
_________________________________________________________________
```

Şekil 31. MobileNetV2 temelli oluşturulan ağ mimarisi outputu

Şekil 25'te girilen hiperparametre değerlerine göre modelde 2652226 adet parametre oluştuğu görülmektedir. Ayrıca oluşturulan modeller arasında en az parametre değerine sahip olan modeldir.

2.4.5. InceptionResNetV2 Temelli Modelin Oluşturulması

Mimarinin temeli, Google tarafından Inception ve ResNet mimarilerinin birleşiminden oluşturulan InceptionResNetV2 mimarisi baz alınarak oluşturulmuştur. Önceki modellerde oluşturulduğu gibi Evrişim katmanından sonra düzleştirme yöntemi için 'Flatten' katmanı modele eklenmiştir. 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele evrişim katmanından sonra eklenmiştir. Modelin son katmanı ise 2 adet 'softmax' aktivasyon fonksiyonu ile

hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu ‘SparseCategoricalCrossentropy’ kullanılmıştır. Metrik ayarları ‘accuracy’ doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu ‘Adam’ olarak seçilmiştir.

```
In [4]: from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.InceptionResNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_InceptionResNetV2 = tf.keras.Sequential()
model_InceptionResNetV2.add(base_model)
model_InceptionResNetV2.add(tf.keras.layers.GlobalAveragePooling2D())

model_InceptionResNetV2.add(tf.keras.layers.Flatten())
model_InceptionResNetV2.add(tf.keras.layers.Dense(256, activation='relu'))
model_InceptionResNetV2.add(tf.keras.layers.Dropout(0.5))
model_InceptionResNetV2.add(tf.keras.layers.Dense(256, activation='relu'))
model_InceptionResNetV2.add(tf.keras.layers.Dropout(0.5))

model_InceptionResNetV2.add(tf.keras.layers.Dense(2, activation='softmax'))

model_InceptionResNetV2.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_InceptionResNetV2.summary()
```

Şekil 32. InceptionResNetV2 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_resnet_v2 (Functi  (None, 5, 5, 1536)       54336736
 onal)

 global_average_pooling2d (G  (None, 1536)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 1536)              0

 dense (Dense)               (None, 256)               393472

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 54,796,514
Trainable params: 459,778
Non-trainable params: 54,336,736
_________________________________________________________________
```

Şekil 33. InceptionResNetV2 temelli oluşturulan ağ mimarisi outputu

Girilen hiperparametre değerlerine göre modelde 54796514 adet parametre oluşmuştur. Ayrıca oluşturulan modeller arasında en fazla parametre değerine sahip olan modeldir.

2.4.6. InceptionV3 Temelli Modelin Oluşturulması

Inception mimarilerinden esinlenilerek Google tarafından üretilen InceptionV3 mimarisi kullanılarak oluşturulan modelde evrişim katmanının hemen ardından düzleştirme işlemi için ‘flatten’ katmanı koyulmuştur. 4 katmanlı bir tamamen bağlı katman 2 katmanı ‘Dropout’ katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele evrişim katmanından sonra eklenmiştir. Modelin son katmanı ise 2 adet ‘softmax’ aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu ‘SparseCategoricalCrossentropy’ kullanılmıştır. Metrik ayarları ‘accuracy’ doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu ‘Adam’ olarak seçilmiştir.

```
In [5]: from tensorflow.keras.layers.experimental import preprocessing

        tf.keras.backend.clear_session()
        input_shape = (height, width, 3)
        base_model = tf.keras.applications.InceptionV3(
            weights='imagenet',
            include_top=False,
            input_shape=input_shape
        )
        base_model.trainable = False

        model_Inceptionv3 = tf.keras.Sequential()
        model_Inceptionv3.add(base_model)
        model_Inceptionv3.add(tf.keras.layers.GlobalAveragePooling2D())

        model_Inceptionv3.add(tf.keras.layers.Flatten())
        model_Inceptionv3.add(tf.keras.layers.Dense(256, activation='relu'))
        model_Inceptionv3.add(tf.keras.layers.Dropout(0.5))
        model_Inceptionv3.add(tf.keras.layers.Dense(256, activation='relu'))
        model_Inceptionv3.add(tf.keras.layers.Dropout(0.5))

        model_Inceptionv3.add(tf.keras.layers.Dense(2, activation='softmax'))

        model_Inceptionv3.compile(loss='SparseCategoricalCrossentropy',
                    optimizer=tf.keras.optimizers.Adam(lr=1e-4),
                    metrics=['acc'])
        model_Inceptionv3.summary()
```

Şekil 34. InceptionV3 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 22,393,634
Trainable params: 590,850
Non-trainable params: 21,802,784
```

Şekil 35. InceptionV3 temelli oluşturulan ağ mimarisi outputu

Girilen hiperparametre değerlerine göre modelde 22393634 adet parametre oluşmuştuğu görülmektedir.

2.4.7. ResNet50 Temelli Modelin Oluşturulması

"Residual Neural Network" ifadesinden gelen ve 2015 yılında Microsoft tarafından üretilen ResNet50 mimarisi baz alınarak model tasarlanmıştır. Benzer şekilde Evrişim katmanından sonra düzleştirme yöntemi için 'Flatten' katmanı modele eklenmiştir. 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele evrişim katmanından sonra eklenmiştir. Modelin son katmanı ise 2 adet 'softmax' aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır. Metrik ayarları 'accuracy' doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu 'Adam' olarak seçilmiştir.

```
In [4]: from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_ResNet50 = tf.keras.Sequential()
model_ResNet50.add(base_model)
model_ResNet50.add(tf.keras.layers.GlobalAveragePooling2D())

model_ResNet50.add(tf.keras.layers.Flatten())
model_ResNet50.add(tf.keras.layers.Dense(256, activation='relu'))
model_ResNet50.add(tf.keras.layers.Dropout(0.5))
model_ResNet50.add(tf.keras.layers.Dense(256, activation='relu'))
model_ResNet50.add(tf.keras.layers.Dropout(0.5))

model_ResNet50.add(tf.keras.layers.Dense(2, activation='softmax'))

model_ResNet50.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
              metrics=['acc'])
model_ResNet50.summary()
```

Şekil 36. ResNet50 temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 24,178,562
Trainable params: 590,850
Non-trainable params: 23,587,712
_________________________________________________________________
```

Şekil 37. ResNet50 temelli oluşturulan ağ mimarisi outputu

Şekil 37'de girilen hiperparametre değerlerine göre modelde 24178562 adet parametre oluştuğu görülmektedir.

2.4.8. Xception Temelli Modelin Oluşturulması

"Extreme Inception" anlamına gelen ve Inception mimarisinin bir varyasyonu olan Xception mimarisi baz alınarak model oluşturulmuştur. Evrişim katmanından sonra düzleştirme yöntemi için 'Flatten' katmanı modele eklenmiştir. 4 katmanlı bir tamamen bağlı katman 2 katmanı 'Dropout' katmanı ve diğer 2 katmanı da 256 adet ReLU fonksiyonu olacak şekilde modele evrişim katmanından sonra eklenmiştir. Modelin son katmanı ise 2 adet 'softmax' aktivasyon fonksiyonu ile hazırlanmıştır. Modelde kategori olarak sınıflandırma problemleri için üretilen hata fonksiyonu 'SparseCategoricalCrossentropy' kullanılmıştır. Metrik ayarları 'accuracy' doğruluk olacak şekilde belirlenip optimizasyon fonksiyonu 'Adam' olarak seçilmiştir.

```
In [4]: from tensorflow.keras.layers.experimental import preprocessing

        tf.keras.backend.clear_session()
        input_shape = (height, width, 3)
        base_model = tf.keras.applications.Xception(
            weights='imagenet',
            include_top=False,
            input_shape=input_shape
        )
        base_model.trainable = False

        model_Xception = tf.keras.Sequential()
        model_Xception.add(base_model)
        model_Xception.add(tf.keras.layers.GlobalAveragePooling2D())

        model_Xception.add(tf.keras.layers.Flatten())
        model_Xception.add(tf.keras.layers.Dense(256, activation='relu'))
        model_Xception.add(tf.keras.layers.Dropout(0.5))
        model_Xception.add(tf.keras.layers.Dense(256, activation='relu'))
        model_Xception.add(tf.keras.layers.Dropout(0.5))

        model_Xception.add(tf.keras.layers.Dense(2, activation='softmax'))

        model_Xception.compile(loss='SparseCategoricalCrossentropy',
                      optimizer=tf.keras.optimizers.Adam(lr=1e-4),
                      metrics=['acc'])
        model_Xception.summary()
```

Şekil 38. Xception temelli oluşturulan ağ mimarisi inputu

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 7, 7, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 21,452,330
Trainable params: 590,850
Non-trainable params: 20,861,480
```

Şekil 39. Xception temelli oluşturulan ağ mimarisi outputu

Girilen hiperparametre değerlerine göre modelde 21452330 adet parametre oluşmuştur.

2.5. Modelin Eğitimi

Model eğitimi için hazırlanan eğitim fonksiyonuna iki adet erken çağırma fonksiyonu *(callback function)* tanımlanmıştır. Bu fonksiyonlardan ilki doğruluk skorunu bir önceki adımdaki doğruluk skoruyla karşılaştırıp en iyi skoru kaydeden 'checkpoint' fonksiyonudur. İkincisi ise modelin aşırı uydurmaya *(overfitting)* maruz kalmaması için hazırlanan erken durdurma *(early stopping)* fonksiyonudur. Eğitim verilerine göre modelin aşırı uydurmasını engellemek için seçilen bu fonksiyon ile doğrulama hata skoru *(validation loss)* 6-7 öğrenim kademesinden sonra azalmak yerine artışa geçiyorsa öğrenim işlemini durdurmayı sağlamaktadır.

Model eğitilirken veri yığınları 60 adım olacak şekilde modele verilmiştir. Ayrıca her adımdan sonra modelin başarısını test etmek için kullanılan doğrulama verileri *(validation data)* için daha önce oluşturulan test verileri kullanılmıştır. Modele veriler karıştırılarak verilmiştir.

Oluşturulan modeller 3 şekilde eğitilmiştir. Tip-1 modeller yani normal veriler ile eğitilecek ağırlıkları şekillenmemiş modeller, Tip-2 modeller yani veri arttırımı yöntemi ile çoğatılmış veriler ile eğitilecek ağırlıkları şekillenmemiş modeller ve Tip-3 modeller yani veri arttırımı yöntemi ile eğitilecek ağırlıkları Tip-1 metodunda belirlenmiş modeller olarak sınıflandırılabilir.

```
In [5]: checkpoint = tf.keras.callbacks.ModelCheckpoint('model/model_vgg16_best.h5', mon
        early = tf.keras.callbacks.EarlyStopping(monitor="val_loss", mode="min",restore_

        callbacks_list = [checkpoint,early]

        history = model_Xception.fit(
                train_generator,
                validation_data = test_generator,
                epochs=60,
                shuffle=True,
                verbose=True,
                callbacks=callbacks_list)
```

Şekil 40. Normal verilerin uygulanacağı model eğitim fonksiyonu

```
In [7]: checkpoint = tf.keras.callbacks.ModelCheckpoint('model/model_vgg16_best.h5', monitor='acc', verbose=1, mode='max',save_best_only=
        early = tf.keras.callbacks.EarlyStopping(monitor="val_loss", mode="min",restore_best_weights=True, patience=7)

        callbacks_list = [checkpoint,early]

        history = model_vgg16.fit(
                aug_train_generator,
                validation_data = test_generator,
                epochs=60,
                shuffle=True,
                verbose=True,
                callbacks=callbacks_list)
```

Şekil 41. Veri arttırımı tekniği uygulanacağı eğitim fonksiyonu

```
In [4]: model_vgg16_t=load_model("model_vgg-.h5")
```

Şekil 42. Tip-3 metodu uygulanacak modelin çağırma inputu

2.6. Adımlar *(Epochs)* ve Grafikler

Eğitim işleminin sonunda her Tip metot için Grafikler ve Adım işlemini gösteren değerler üretilmiştir.

2.6.1. VGG16 Temelli Modelin Grafik ve Adım Sonuçları

VGG16 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

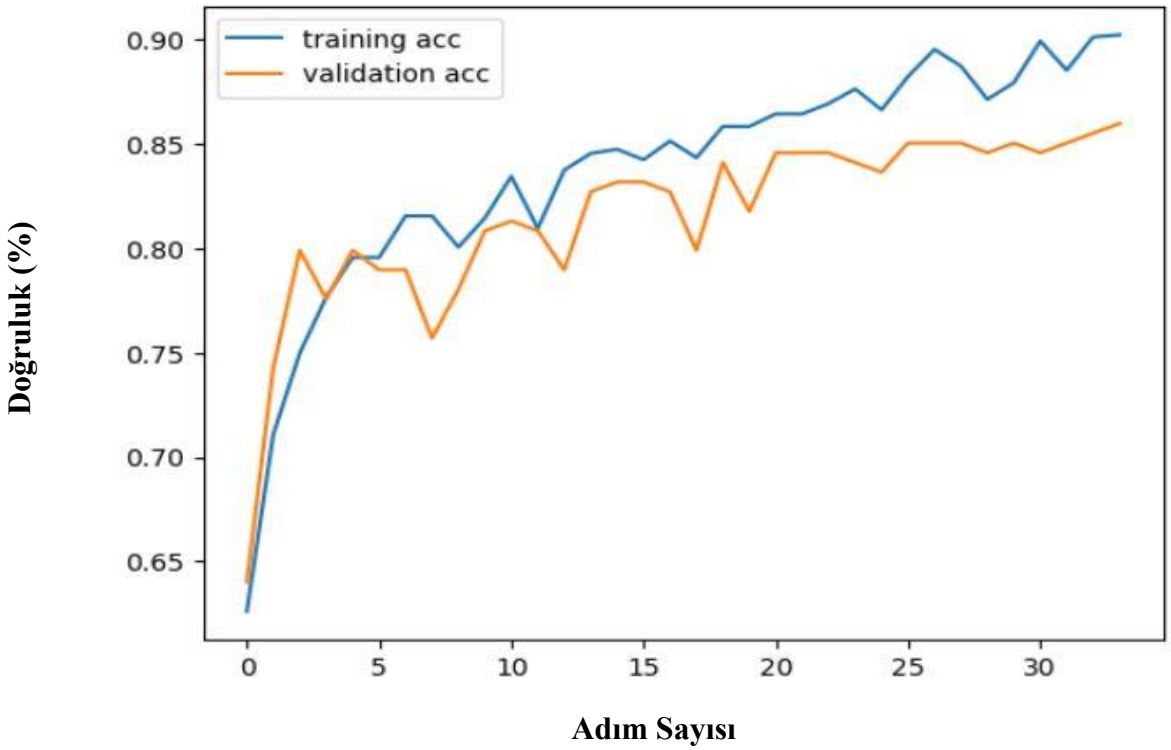
VGG16 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen VGG16 Tip – 1 modeli 34 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.2516 - acc: 0.8794
Epoch 30: acc did not improve from 0.89531
16/16 [==============================] - 155s 10s/step - loss: 0.2516 - acc: 0.8794 - val_loss: 0.3412 - val_acc: 0.8505
Epoch 31/60
16/16 [==============================] - ETA: 0s - loss: 0.2425 - acc: 0.8993
Epoch 31: acc improved from 0.89531 to 0.89930, saving model to model\model_vgg16_best.h5
16/16 [==============================] - 152s 10s/step - loss: 0.2425 - acc: 0.8993 - val_loss: 0.3450 - val_acc: 0.8458
Epoch 32/60
16/16 [==============================] - ETA: 0s - loss: 0.2385 - acc: 0.8853
Epoch 32: acc did not improve from 0.89930
16/16 [==============================] - 152s 10s/step - loss: 0.2385 - acc: 0.8853 - val_loss: 0.3419 - val_acc: 0.8505
Epoch 33/60
16/16 [==============================] - ETA: 0s - loss: 0.2272 - acc: 0.9013
Epoch 33: acc improved from 0.89930 to 0.90130, saving model to model\model_vgg16_best.h5
16/16 [==============================] - 152s 10s/step - loss: 0.2272 - acc: 0.9013 - val_loss: 0.3512 - val_acc: 0.8551
Epoch 34/60
16/16 [==============================] - ETA: 0s - loss: 0.2311 - acc: 0.9023
Epoch 34: acc improved from 0.90130 to 0.90229, saving model to model\model_vgg16_best.h5
16/16 [==============================] - 151s 10s/step - loss: 0.2311 - acc: 0.9023 - val_loss: 0.3299 - val_acc: 0.8598
```

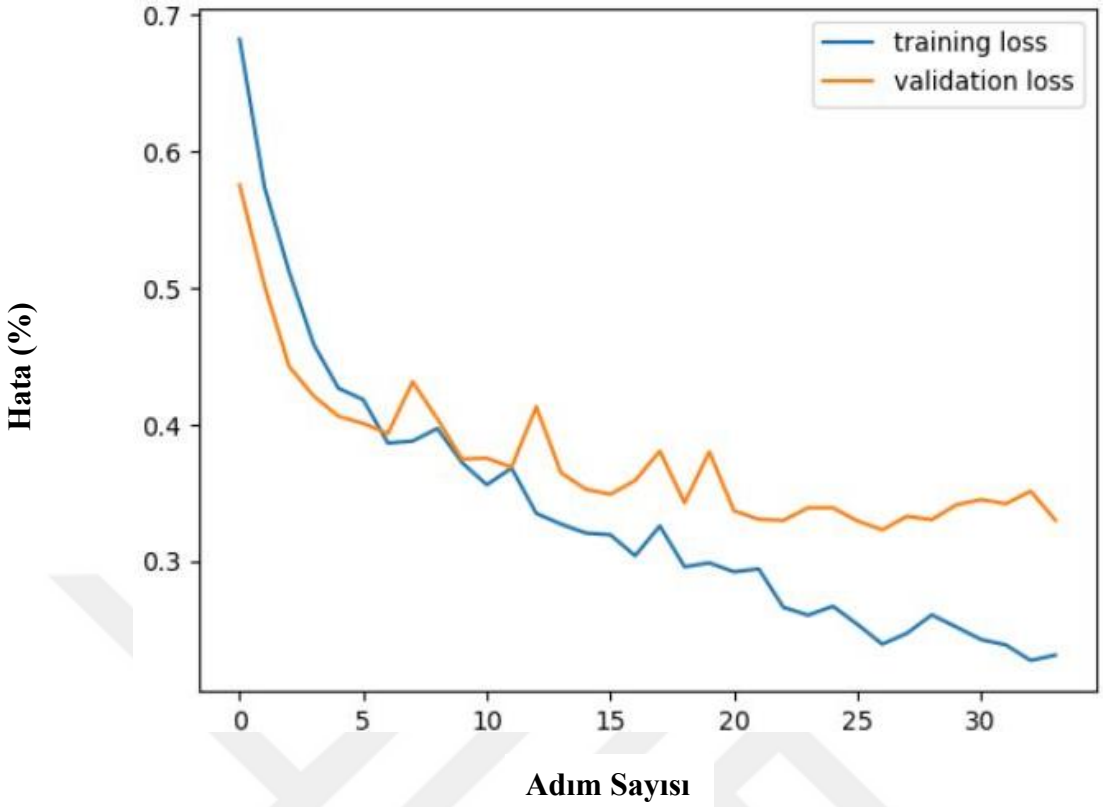
Şekil 43. VGG16 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 44. VGG16 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 44’teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 45. VGG16 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 45'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

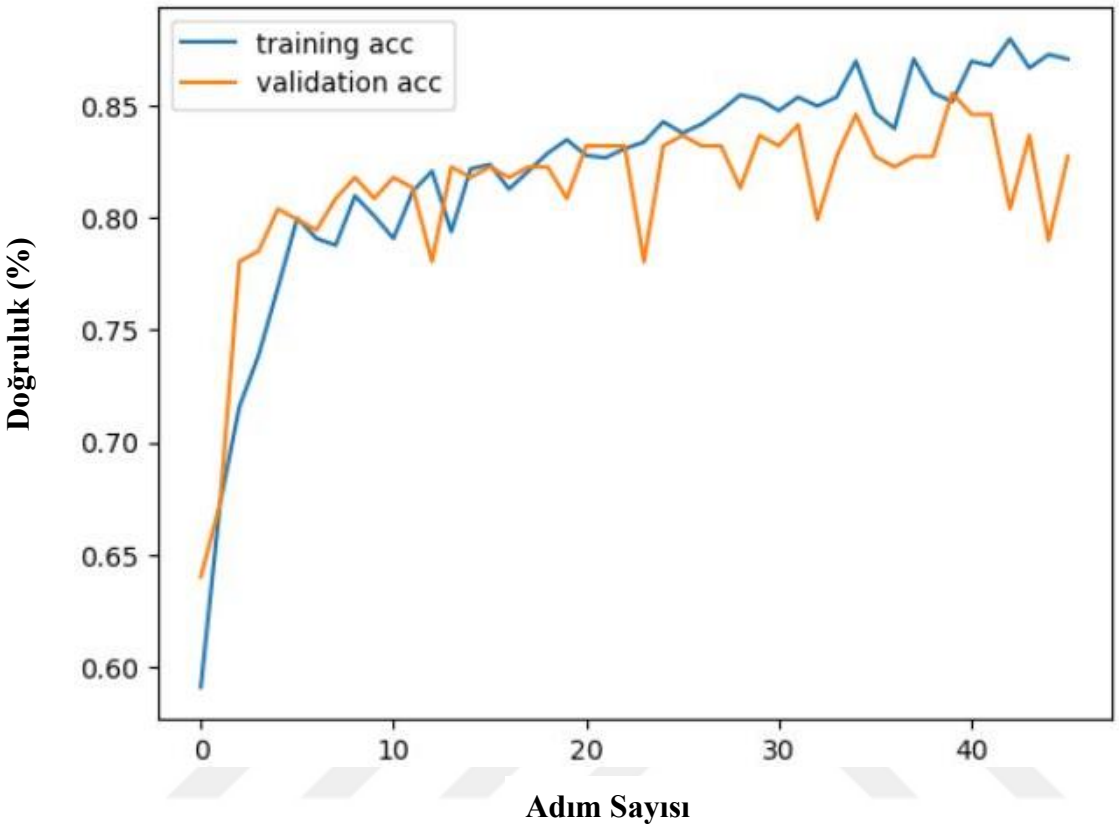
VGG16 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen VGG16 Tip – 2 modeli 46 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.2853 - acc: 0.8674
Epoch 42: acc did not improve from 0.87039
16/16 [==============================] - 158s 10s/step - loss: 0.2853 - acc: 0.8674 - val_loss: 0.3711 - val_acc: 0.8458
Epoch 43/60
16/16 [==============================] - ETA: 0s - loss: 0.2765 - acc: 0.8794
Epoch 43: acc improved from 0.87039 to 0.87936, saving model to model\vgg16_best.h5
16/16 [==============================] - 158s 10s/step - loss: 0.2765 - acc: 0.8794 - val_loss: 0.3538 - val_acc: 0.8037
Epoch 44/60
16/16 [==============================] - ETA: 0s - loss: 0.2895 - acc: 0.8664
Epoch 44: acc did not improve from 0.87936
16/16 [==============================] - 160s 10s/step - loss: 0.2895 - acc: 0.8664 - val_loss: 0.3711 - val_acc: 0.8364
Epoch 45/60
16/16 [==============================] - ETA: 0s - loss: 0.2782 - acc: 0.8724
Epoch 45: acc did not improve from 0.87936
16/16 [==============================] - 159s 10s/step - loss: 0.2782 - acc: 0.8724 - val_loss: 0.3996 - val_acc: 0.7897
Epoch 46/60
16/16 [==============================] - ETA: 0s - loss: 0.2850 - acc: 0.8704
Epoch 46: acc did not improve from 0.87936
16/16 [==============================] - 159s 10s/step - loss: 0.2850 - acc: 0.8704 - val_loss: 0.3440 - val_acc: 0.8271
```

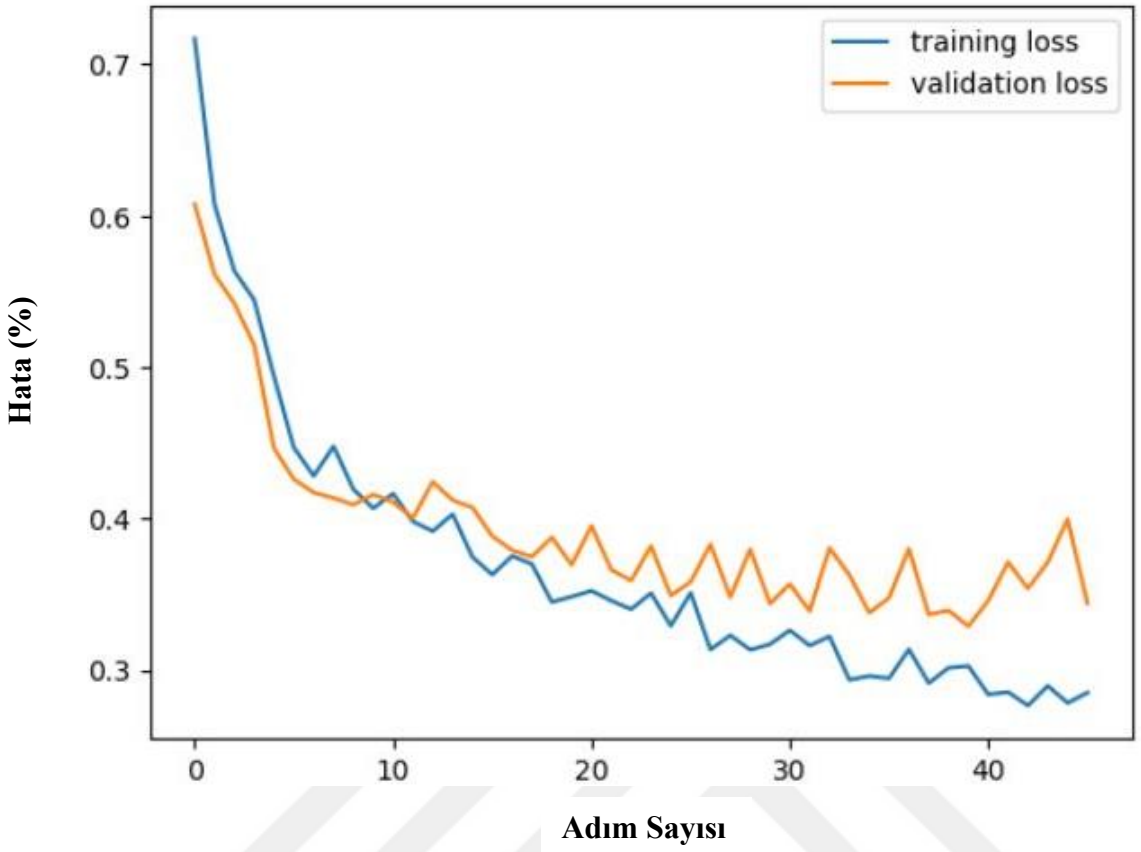
Şekil 46. VGG16 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 47. VGG16 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 47’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir. Grafikte tespit edildiği üzere 40. Adımdan sonra doğruluk skorları birbirinden uzaklaşmaya başlamaktadır.



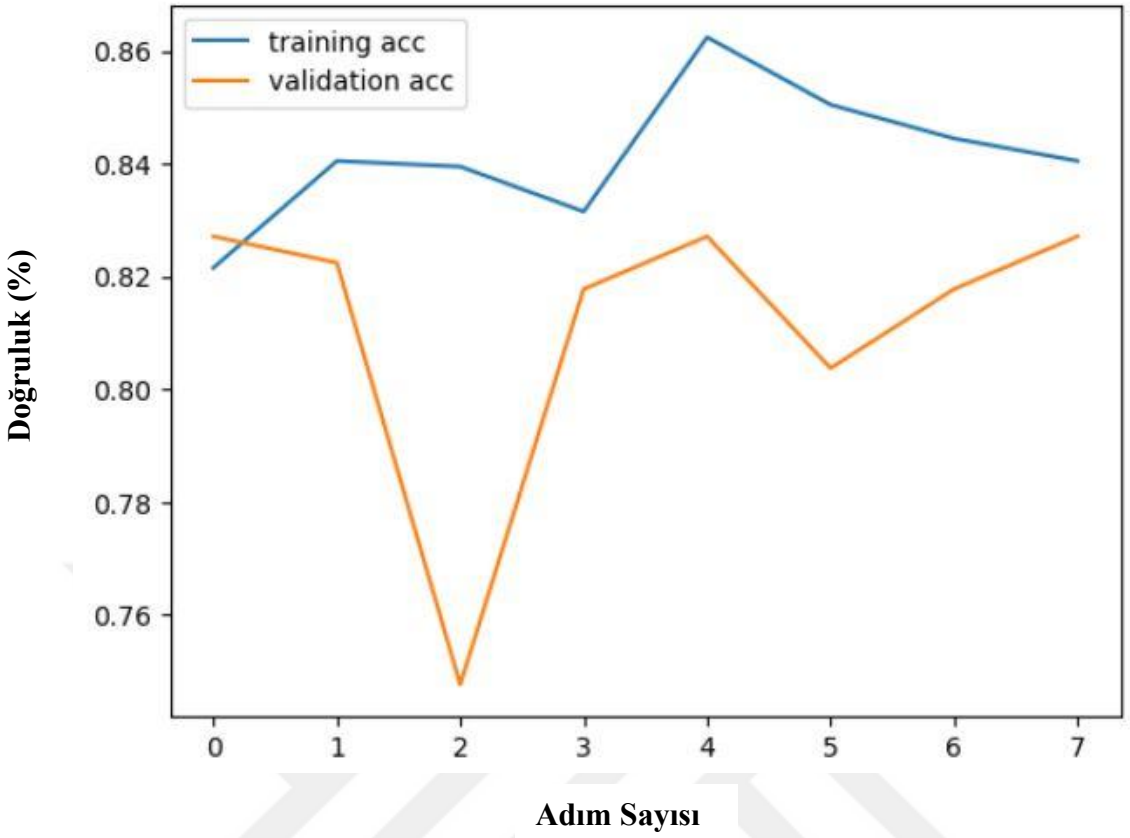
Şekil 48. VGG16 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 48'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

VGG16 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

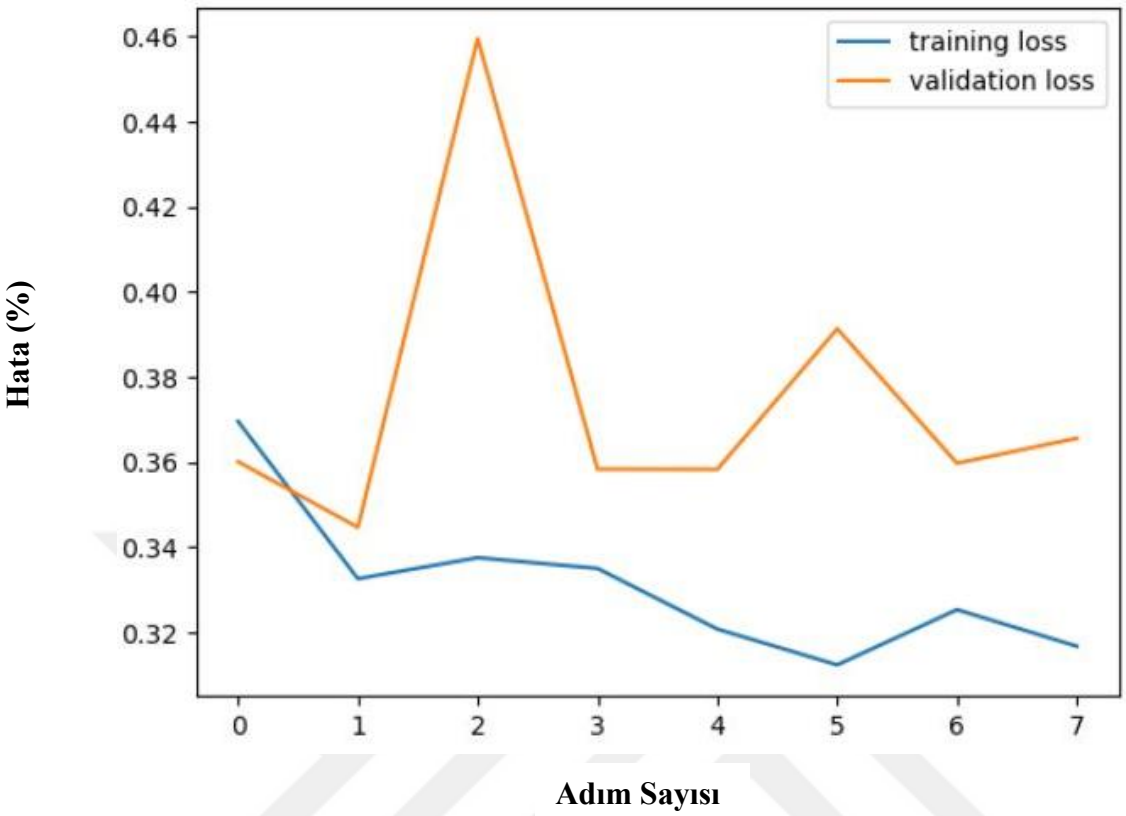
Eğitilen VGG16 Tip – 3 modeli 7 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu 'doğruluk' ve 'hata' grafiği elde edilmiştir.



Şekil 49. VGG16 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 49'daki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir. Grafikte tespit edildiği üzere önceden eğitilmiş olan modelde ufak çaplı değişiklikler gözlenmiştir.



Şekil 50. VGG16 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 50'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.2. VGG19 Temelli Modelin Grafik ve Adım Sonuçları

VGG19 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

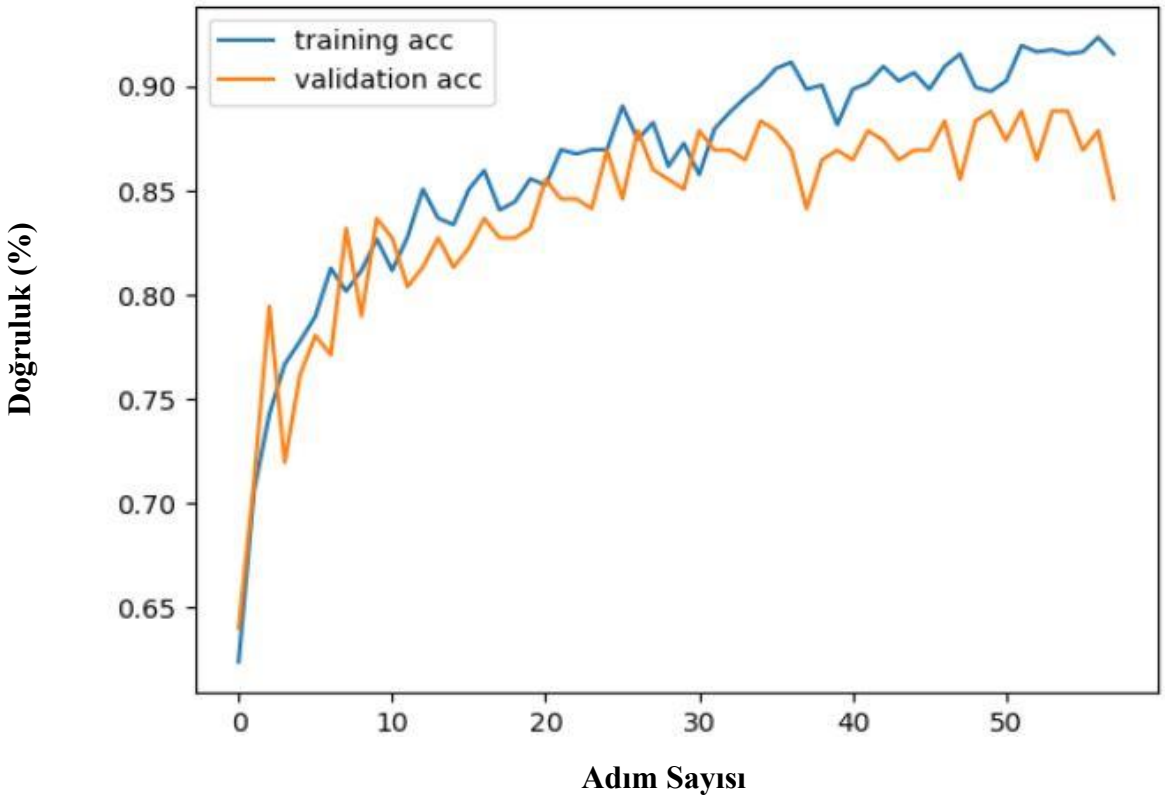
VGG19 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen VGG19 Tip – 1 modeli 58 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur. 60 olarak belirlenen adım sayısının tamamlanmasına 2 adım kala fonksiyon durdurulmuştur. Bu sonuç göze alındığında oluşturulan modelden başarılı skorlar beklenmektedir.

```
16/16 [==============================] - ETA: 0s - loss: 0.1878 - acc: 0.9172
Epoch 54: acc did not improve from 0.91924
16/16 [==============================] - 207s 13s/step - loss: 0.1878 - acc: 0.9172 - val_loss: 0.2661 - val_acc: 0.8879
Epoch 55/60
16/16 [==============================] - ETA: 0s - loss: 0.1912 - acc: 0.9153
Epoch 55: acc did not improve from 0.91924
16/16 [==============================] - 201s 13s/step - loss: 0.1912 - acc: 0.9153 - val_loss: 0.2683 - val_acc: 0.8879
Epoch 56/60
16/16 [==============================] - ETA: 0s - loss: 0.1837 - acc: 0.9163
Epoch 56: acc did not improve from 0.91924
16/16 [==============================] - 201s 13s/step - loss: 0.1837 - acc: 0.9163 - val_loss: 0.2719 - val_acc: 0.8692
Epoch 57/60
16/16 [==============================] - ETA: 0s - loss: 0.1782 - acc: 0.9232
Epoch 57: acc improved from 0.91924 to 0.92323, saving model to model\model_vgg19.h5
16/16 [==============================] - 201s 13s/step - loss: 0.1782 - acc: 0.9232 - val_loss: 0.2644 - val_acc: 0.8785
Epoch 58/60
16/16 [==============================] - ETA: 0s - loss: 0.1914 - acc: 0.9153
Epoch 58: acc did not improve from 0.92323
16/16 [==============================] - 194s 12s/step - loss: 0.1914 - acc: 0.9153 - val_loss: 0.2962 - val_acc: 0.8458
```

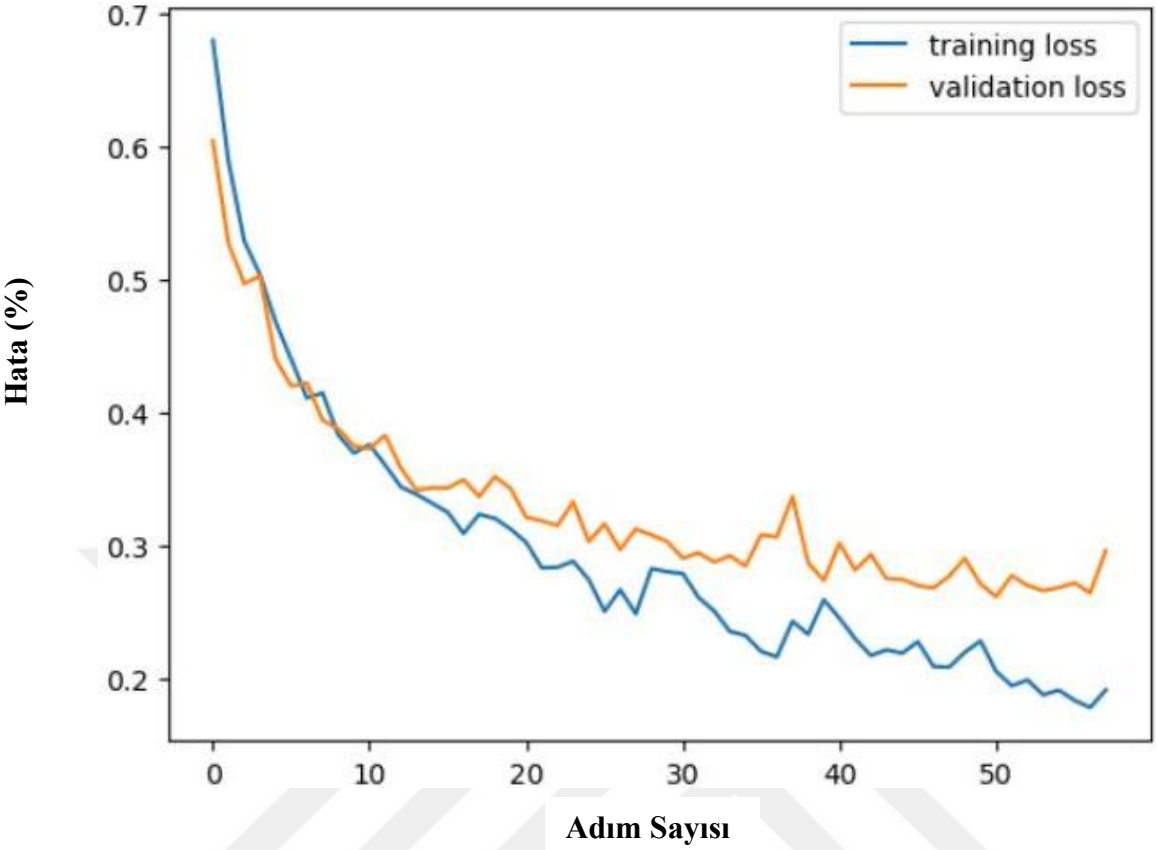
Şekil 51. VGG19 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 52. VGG19 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 52’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 53. VGG19 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 53'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

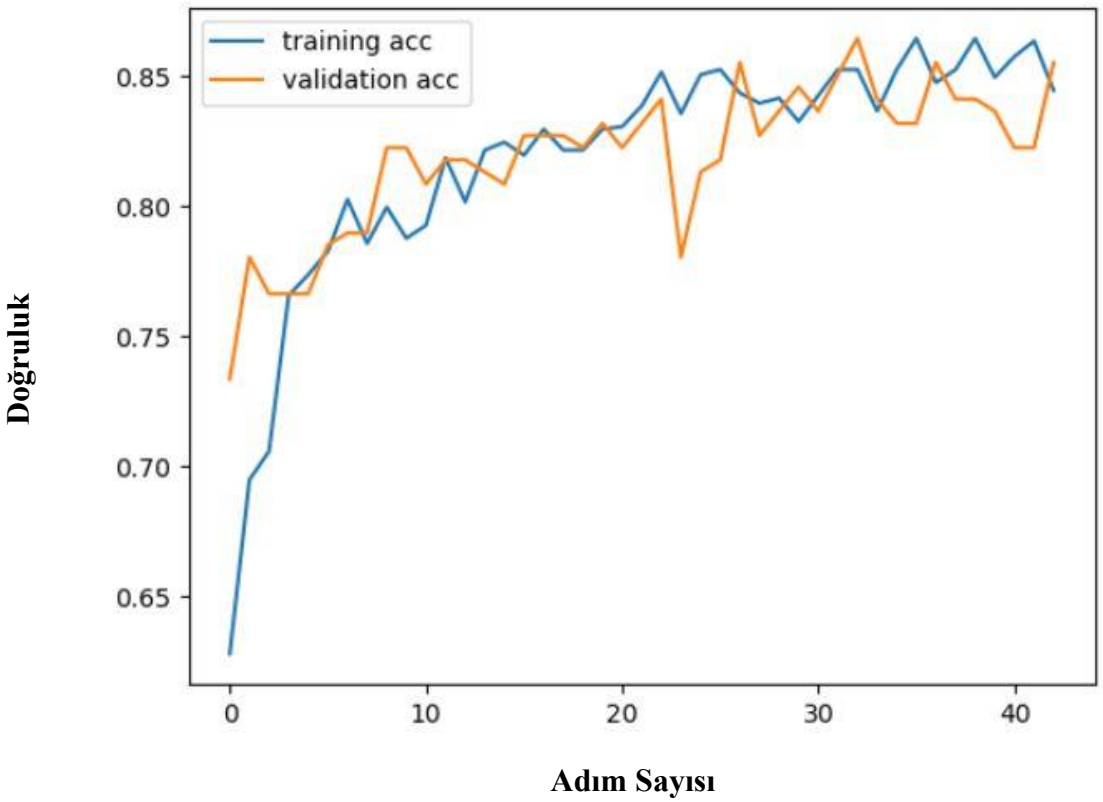
VGG19 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen VGG19 Tip – 2 modeli 43 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.3034 - acc: 0.8644
Epoch 39: acc did not improve from 0.86441
16/16 [==============================] - 201s 13s/step - loss: 0.3034 - acc: 0.8644 - val_loss: 0.3256 - val_acc: 0.8411
Epoch 40/60
16/16 [==============================] - ETA: 0s - loss: 0.3016 - acc: 0.8495
Epoch 40: acc did not improve from 0.86441
16/16 [==============================] - 198s 12s/step - loss: 0.3016 - acc: 0.8495 - val_loss: 0.3546 - val_acc: 0.8364
Epoch 41/60
16/16 [==============================] - ETA: 0s - loss: 0.3016 - acc: 0.8574
Epoch 41: acc did not improve from 0.86441
16/16 [==============================] - 201s 13s/step - loss: 0.3016 - acc: 0.8574 - val_loss: 0.3656 - val_acc: 0.8224
Epoch 42/60
16/16 [==============================] - ETA: 0s - loss: 0.3085 - acc: 0.8634
Epoch 42: acc did not improve from 0.86441
16/16 [==============================] - 202s 13s/step - loss: 0.3085 - acc: 0.8634 - val_loss: 0.3357 - val_acc: 0.8224
Epoch 43/60
16/16 [==============================] - ETA: 0s - loss: 0.3212 - acc: 0.8445
Epoch 43: acc did not improve from 0.86441
16/16 [==============================] - 199s 12s/step - loss: 0.3212 - acc: 0.8445 - val_loss: 0.3261 - val_acc: 0.8551
```

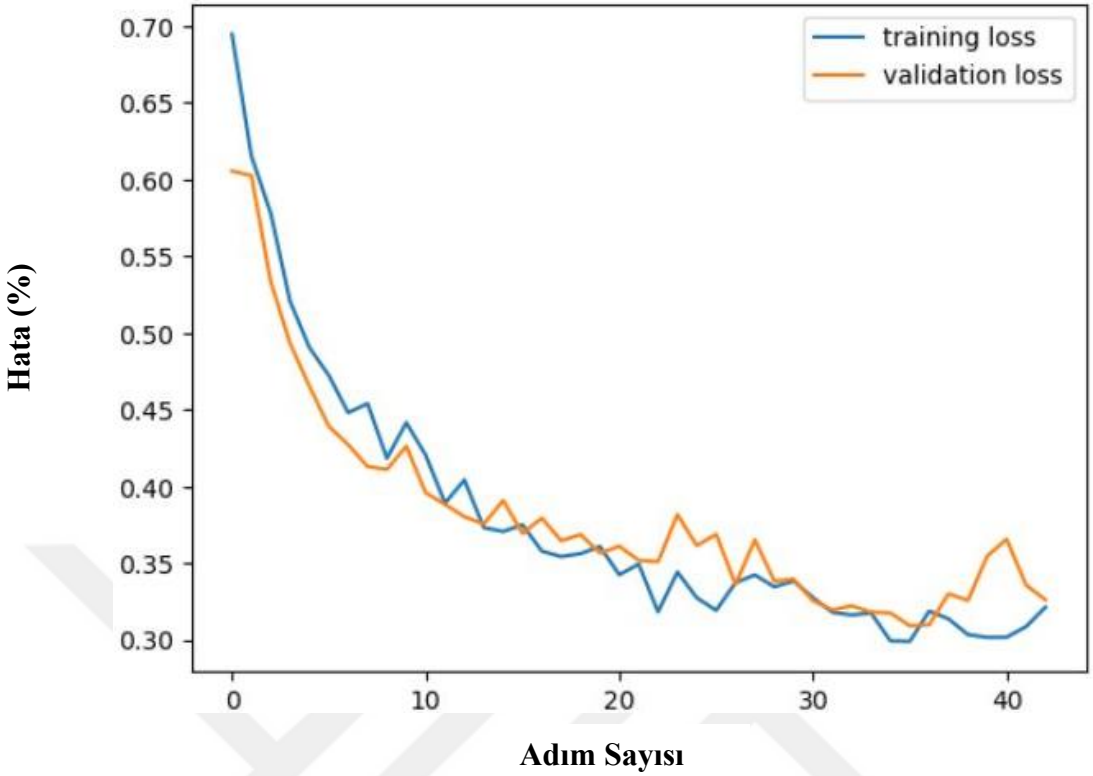
Şekil 54. VGG19 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 55. VGG19 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 55’teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 56. VGG19 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 56'daki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir. Şekil 56'daki grafikten görüldüğü üzere modelin eğitimi 7 adım boyunca doğrulama hata skoru, eğitim skorundan daha büyük bir değer olduğu için 43. adımda değerler birbirine yaklaşsa bile durdurma işlemi gerçekleşmiştir.

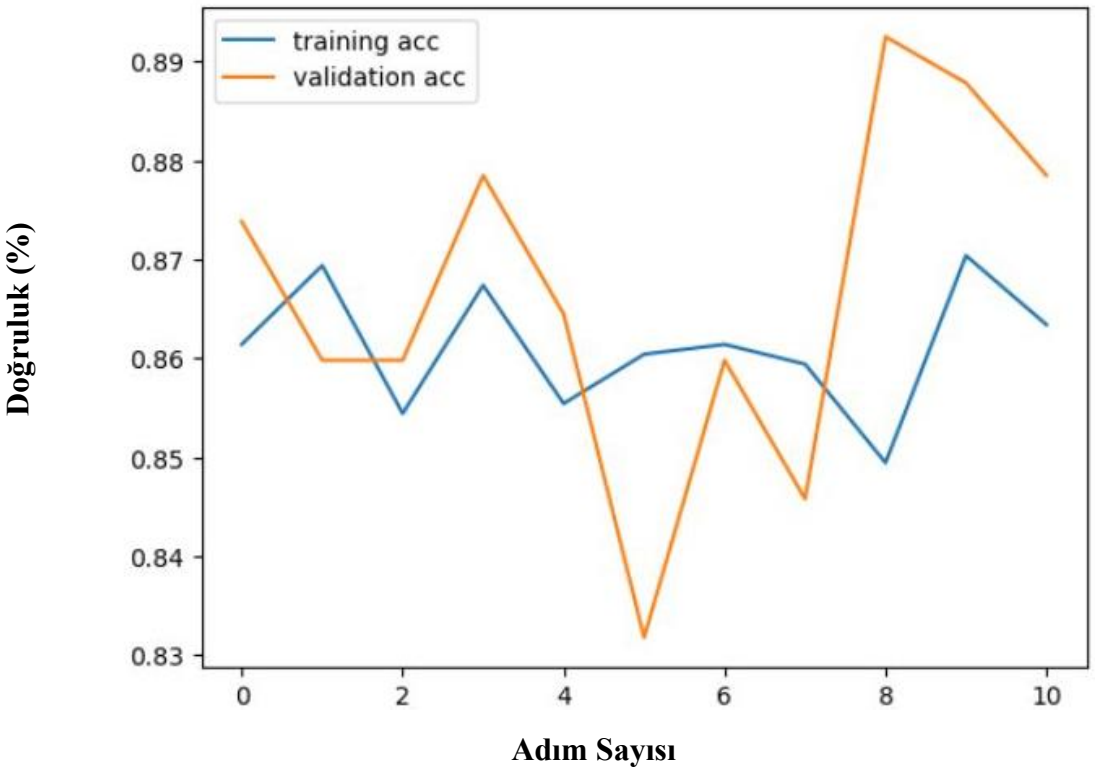
VGG19 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen VGG19 Tip – 3 modeli 11 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 4/60
16/16 [==============================] - ETA: 0s - loss: 0.2946 - acc: 0.8674
Epoch 4: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.2946 - acc: 0.8674 - val_loss: 0.2620 - val_acc: 0.8785
Epoch 5/60
16/16 [==============================] - ETA: 0s - loss: 0.2893 - acc: 0.8554
Epoch 5: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.2893 - acc: 0.8554 - val_loss: 0.2771 - val_acc: 0.8645
Epoch 6/60
16/16 [==============================] - ETA: 0s - loss: 0.2990 - acc: 0.8604
Epoch 6: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.2990 - acc: 0.8604 - val_loss: 0.3390 - val_acc: 0.8318
Epoch 7/60
16/16 [==============================] - ETA: 0s - loss: 0.3014 - acc: 0.8614
Epoch 7: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.3014 - acc: 0.8614 - val_loss: 0.2899 - val_acc: 0.8598
Epoch 8/60
16/16 [==============================] - ETA: 0s - loss: 0.2902 - acc: 0.8594
Epoch 8: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.2902 - acc: 0.8594 - val_loss: 0.3170 - val_acc: 0.8458
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.3143 - acc: 0.8495
Epoch 9: acc did not improve from 0.86939
16/16 [==============================] - 189s 12s/step - loss: 0.3143 - acc: 0.8495 - val_loss: 0.2657 - val_acc: 0.8925
Epoch 10/60
16/16 [==============================] - ETA: 0s - loss: 0.2858 - acc: 0.8704
Epoch 10: acc improved from 0.86939 to 0.87039, saving model to model\model_vgg19.h5
16/16 [==============================] - 190s 12s/step - loss: 0.2858 - acc: 0.8704 - val_loss: 0.2718 - val_acc: 0.8879
Epoch 11/60
16/16 [==============================] - ETA: 0s - loss: 0.2845 - acc: 0.8634
Epoch 11: acc did not improve from 0.87039
16/16 [==============================] - 189s 12s/step - loss: 0.2845 - acc: 0.8634 - val_loss: 0.2944 - val_acc: 0.8785
```

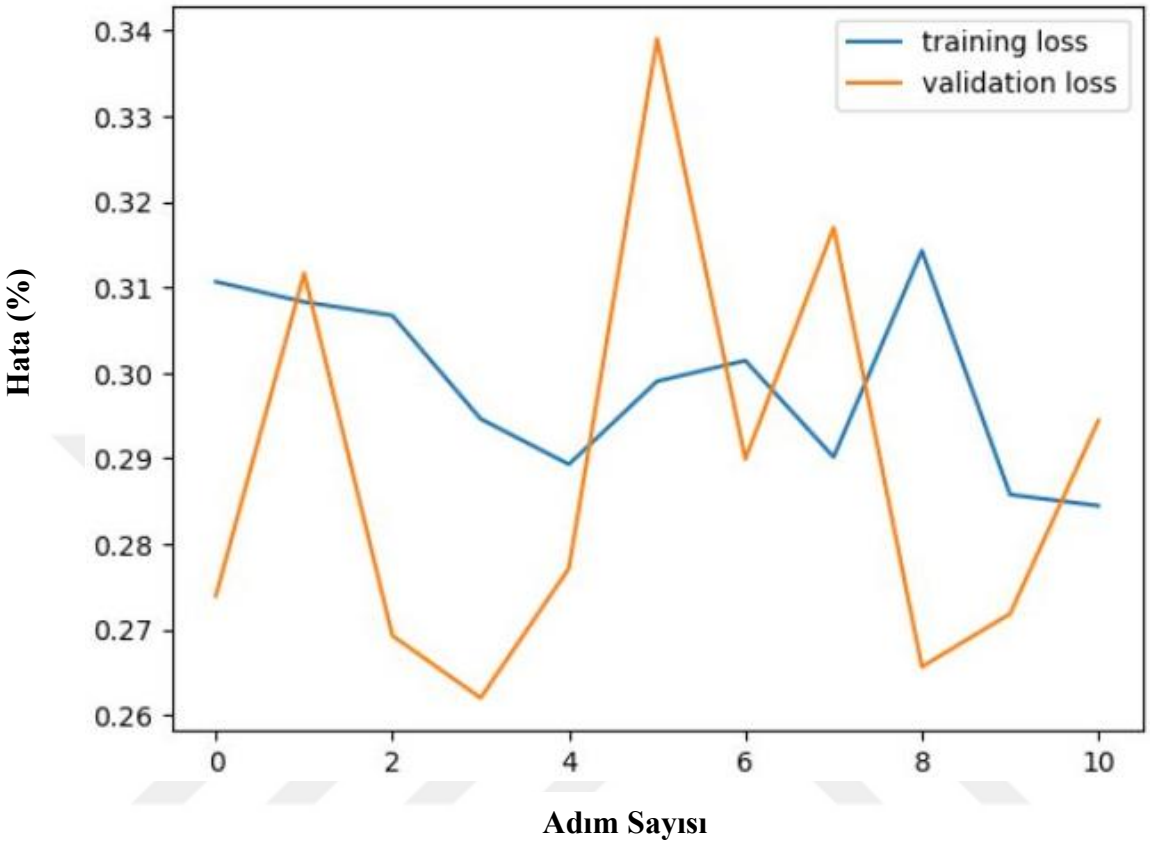
Şekil 57. VGG19 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 58. VGG19 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 58’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 59. VGG19 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 59’daki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.3. ResNet50V2 Temelli Modelin Grafik ve Adım Sonuçları

ResNet50V2 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

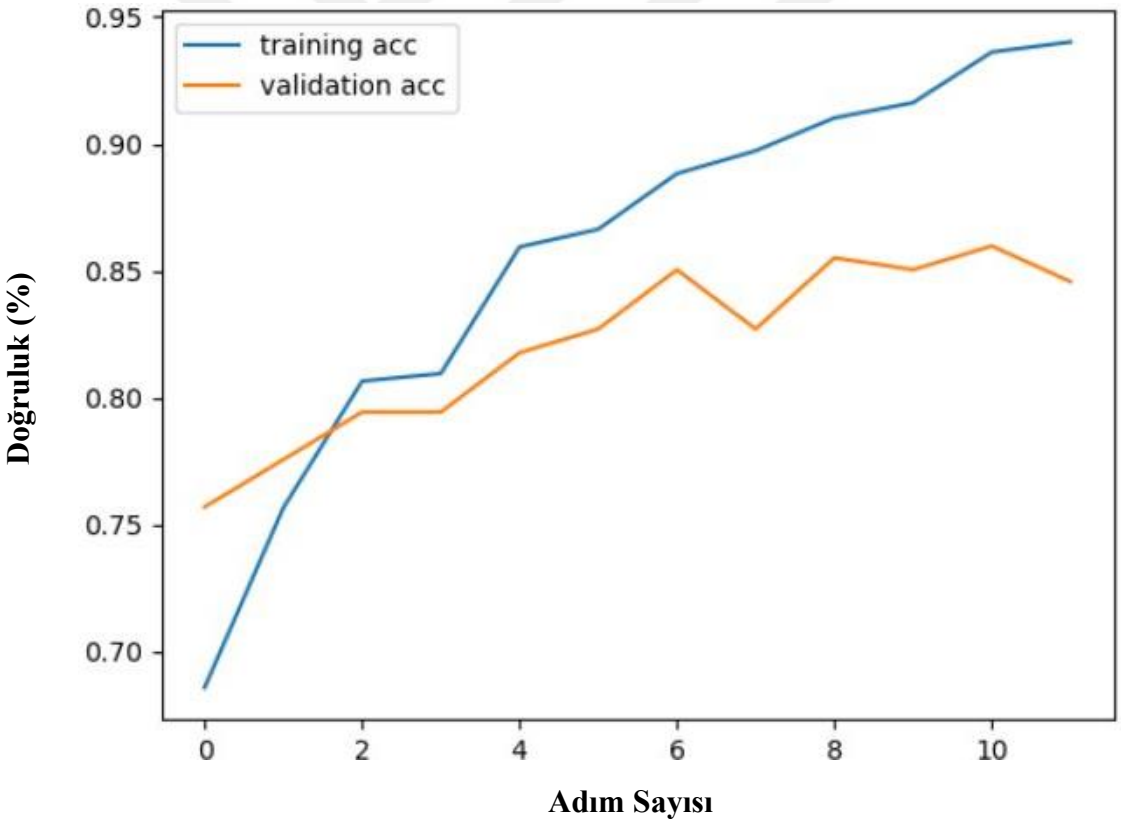
ResNet50V2 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50V2 Tip – 1 modeli 12 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur. Modeli eğitilirken adımlar için harcadığı sürenin diğer modellere oranla daha düşük olduğu gözlenmiştir.

```
Epoch 7: acc improved from 0.86640 to 0.88833, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 65s 4s/step - loss: 0.2416 - acc: 0.8883 - val_loss: 0.3642 - val_acc: 0.8505
Epoch 8/60
16/16 [==============================] - ETA: 0s - loss: 0.2172 - acc: 0.8973
Epoch 8: acc improved from 0.88833 to 0.89731, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 65s 4s/step - loss: 0.2172 - acc: 0.8973 - val_loss: 0.4357 - val_acc: 0.8271
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.2073 - acc: 0.9103
Epoch 9: acc improved from 0.89731 to 0.91027, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 67s 4s/step - loss: 0.2073 - acc: 0.9103 - val_loss: 0.3779 - val_acc: 0.8551
Epoch 10/60
16/16 [==============================] - ETA: 0s - loss: 0.1964 - acc: 0.9163
Epoch 10: acc improved from 0.91027 to 0.91625, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 68s 4s/step - loss: 0.1964 - acc: 0.9163 - val_loss: 0.3859 - val_acc: 0.8505
Epoch 11/60
16/16 [==============================] - ETA: 0s - loss: 0.1531 - acc: 0.9362
Epoch 11: acc improved from 0.91625 to 0.93619, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 68s 4s/step - loss: 0.1531 - acc: 0.9362 - val_loss: 0.3936 - val_acc: 0.8598
Epoch 12/60
16/16 [==============================] - ETA: 0s - loss: 0.1442 - acc: 0.9402
Epoch 12: acc improved from 0.93619 to 0.94018, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 67s 4s/step - loss: 0.1442 - acc: 0.9402 - val_loss: 0.3982 - val_acc: 0.8458
```

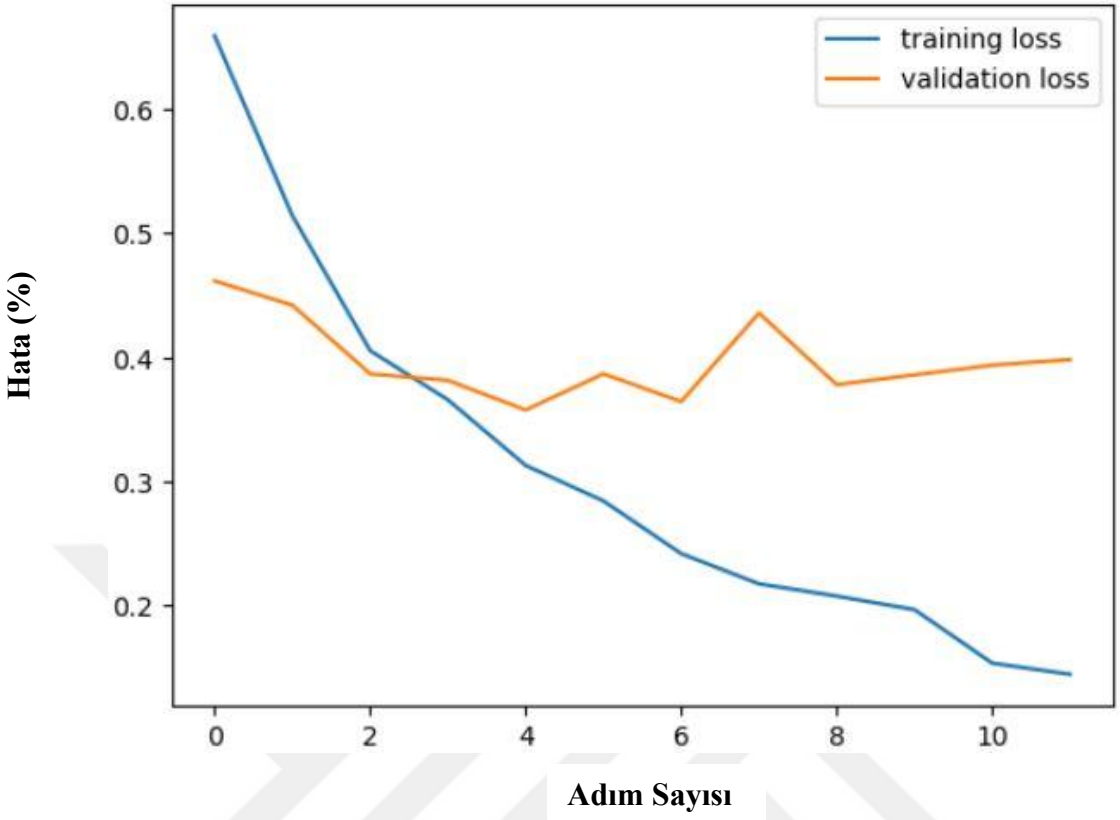
Şekil 60. ResNet50V2 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 61. ResNet50V2 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 61’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 62. ResNet50V2 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 62'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

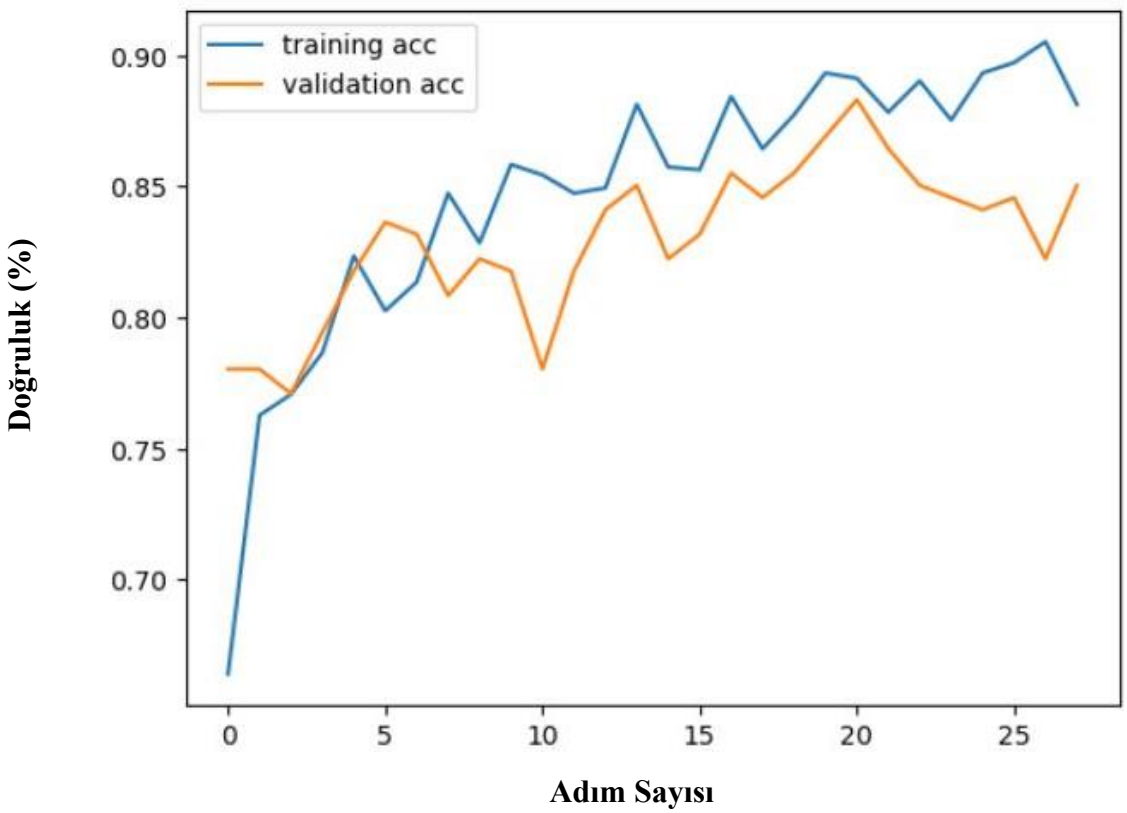
ResNet50V2 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50V2 Tip – 2 modeli 28 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.2547 - acc: 0.8754
Epoch 24: acc did not improve from 0.89332
16/16 [==============================] - 70s 4s/step - loss: 0.2547 - acc: 0.8754 - val_loss: 0.3635 - val_acc: 0.8458
Epoch 25/60
16/16 [==============================] - ETA: 0s - loss: 0.2412 - acc: 0.8933
Epoch 25: acc did not improve from 0.89332
16/16 [==============================] - 70s 4s/step - loss: 0.2412 - acc: 0.8933 - val_loss: 0.3708 - val_acc: 0.8411
Epoch 26/60
16/16 [==============================] - ETA: 0s - loss: 0.2240 - acc: 0.8973
Epoch 26: acc improved from 0.89332 to 0.89731, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 71s 4s/step - loss: 0.2240 - acc: 0.8973 - val_loss: 0.3857 - val_acc: 0.8458
Epoch 27/60
16/16 [==============================] - ETA: 0s - loss: 0.2180 - acc: 0.9053
Epoch 27: acc improved from 0.89731 to 0.90528, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 72s 4s/step - loss: 0.2180 - acc: 0.9053 - val_loss: 0.4222 - val_acc: 0.8224
Epoch 28/60
16/16 [==============================] - ETA: 0s - loss: 0.2540 - acc: 0.8814
Epoch 28: acc did not improve from 0.90528
16/16 [==============================] - 73s 5s/step - loss: 0.2540 - acc: 0.8814 - val_loss: 0.3401 - val_acc: 0.8505
```

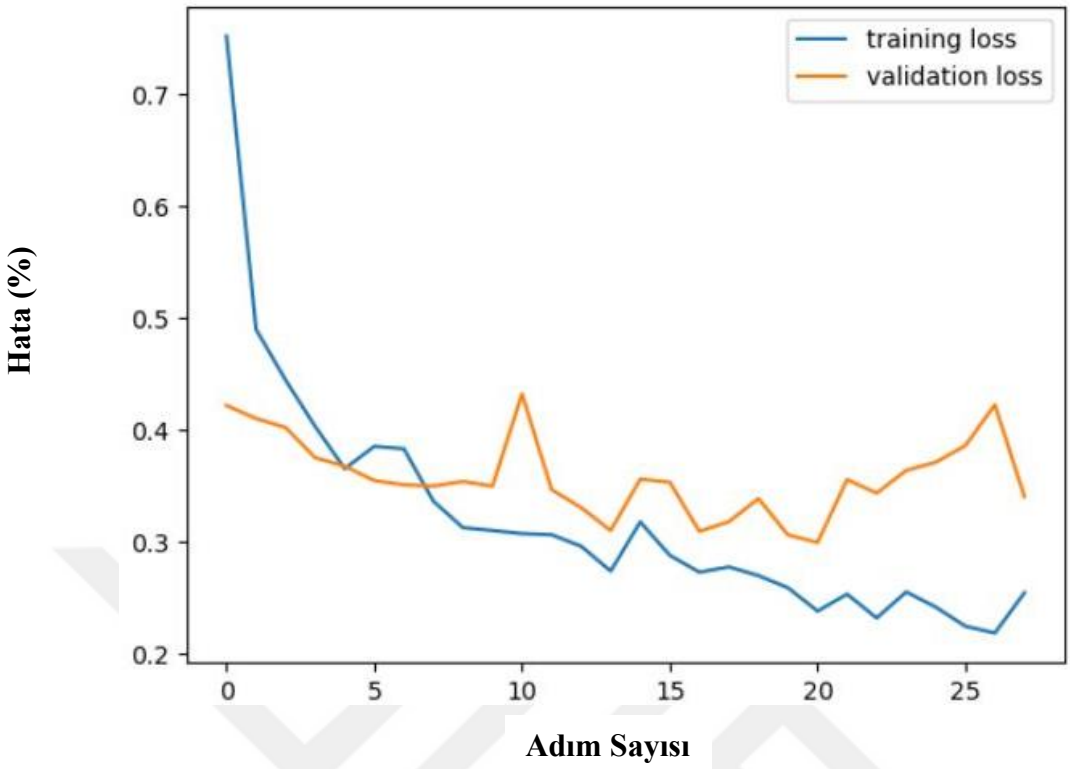
Şekil 63. ResNet50V2 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 64. ResNet50V2 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 64’teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 65. ResNet50V2 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 65'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

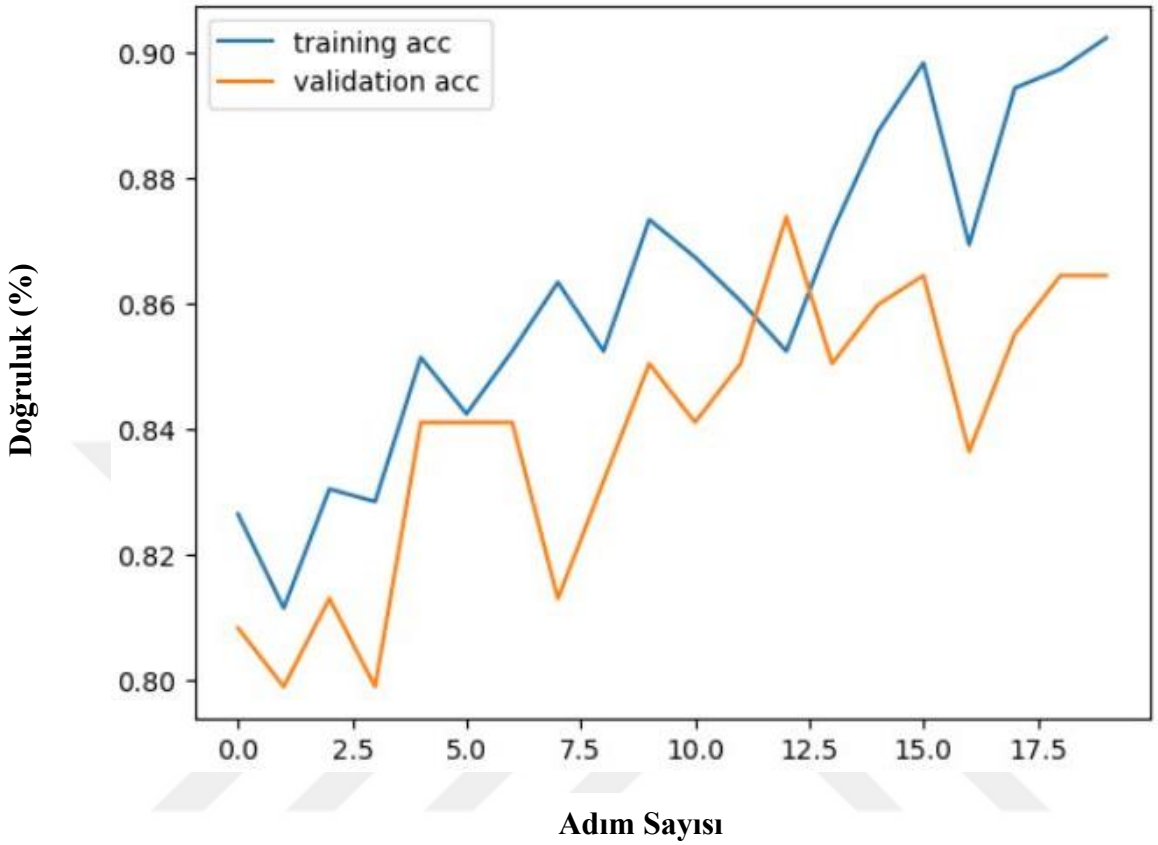
ResNet50V2 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50V2 Tip – 3 modeli 20 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 17/60
16/16 [==============================] - ETA: 0s - loss: 0.2571 - acc: 0.8694
Epoch 17: acc did not improve from 0.89831
16/16 [==============================] - 69s 4s/step - loss: 0.2571 - acc: 0.8694 - val_loss: 0.3368 - val_acc: 0.8364
Epoch 18/60
16/16 [==============================] - ETA: 0s - loss: 0.2352 - acc: 0.8943
Epoch 18: acc did not improve from 0.89831
16/16 [==============================] - 67s 4s/step - loss: 0.2352 - acc: 0.8943 - val_loss: 0.3265 - val_acc: 0.8551
Epoch 19/60
16/16 [==============================] - ETA: 0s - loss: 0.2536 - acc: 0.8973
Epoch 19: acc did not improve from 0.89831
16/16 [==============================] - 70s 4s/step - loss: 0.2536 - acc: 0.8973 - val_loss: 0.3474 - val_acc: 0.8645
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.2252 - acc: 0.9023
Epoch 20: acc improved from 0.89831 to 0.90229, saving model to model\model_ResNet50V2_best.h5
16/16 [==============================] - 70s 4s/step - loss: 0.2252 - acc: 0.9023 - val_loss: 0.4030 - val_acc: 0.8645
```

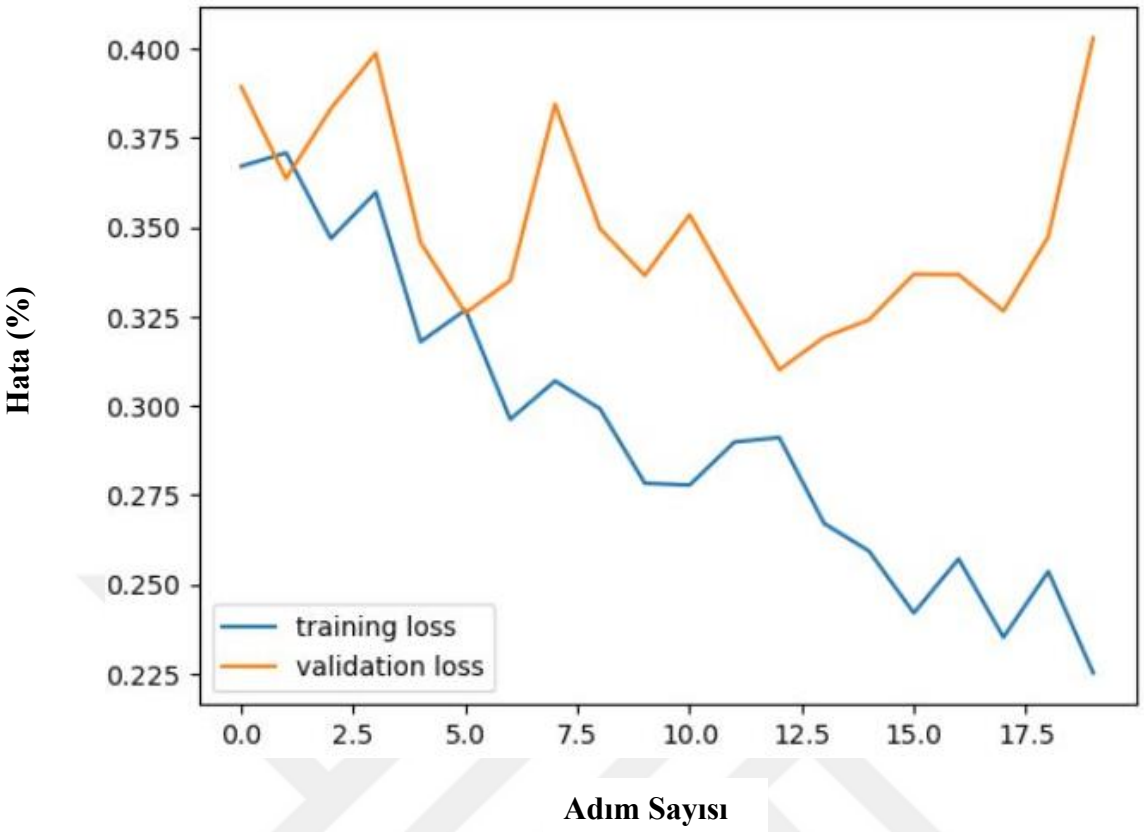
Şekil 66. ResNet50V2 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 67. ResNet50V2 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 67’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 68. ResNet50V2 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 68'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.4. MobileNetV2 Temelli Modelin Grafik ve Adım Sonuçları

MobileNetV2 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

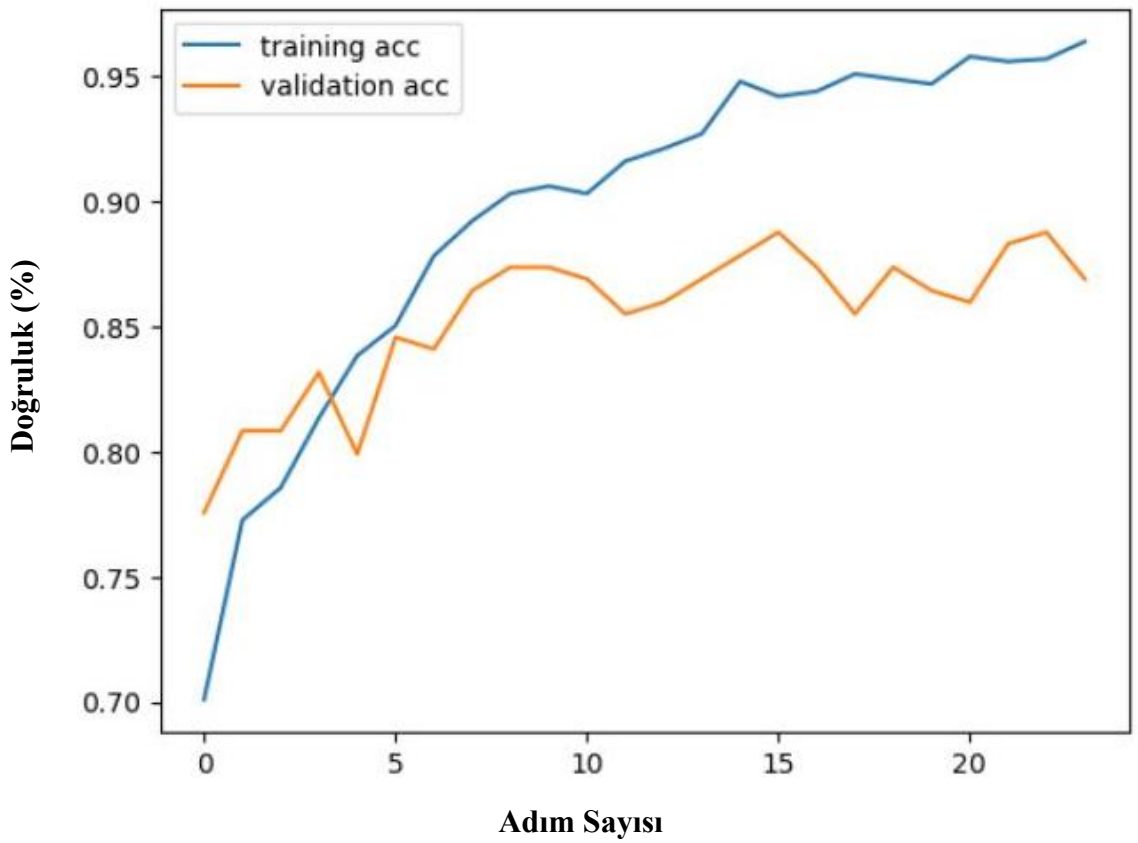
MobileNetV2 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen MobileNetV2 Tip – 1 modeli 24 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.1246 - acc: 0.9472
Epoch 20: acc did not improve from 0.95115
16/16 [==============================] - 26s 2s/step - loss: 0.1246 - acc: 0.9472 - val_loss: 0.3836 - val_acc: 0.8645
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.0999 - acc: 0.9581
Epoch 21: acc improved from 0.95115 to 0.95813, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 27s 2s/step - loss: 0.0999 - acc: 0.9581 - val_loss: 0.3313 - val_acc: 0.8598
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.1056 - acc: 0.9561
Epoch 22: acc did not improve from 0.95813
16/16 [==============================] - 27s 2s/step - loss: 0.1056 - acc: 0.9561 - val_loss: 0.2964 - val_acc: 0.8832
Epoch 23/60
16/16 [==============================] - ETA: 0s - loss: 0.1102 - acc: 0.9571
Epoch 23: acc did not improve from 0.95813
16/16 [==============================] - 26s 2s/step - loss: 0.1102 - acc: 0.9571 - val_loss: 0.2854 - val_acc: 0.8879
Epoch 24/60
16/16 [==============================] - ETA: 0s - loss: 0.0853 - acc: 0.9641
Epoch 24: acc improved from 0.95813 to 0.96411, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 27s 2s/step - loss: 0.0853 - acc: 0.9641 - val_loss: 0.3588 - val_acc: 0.8692
```

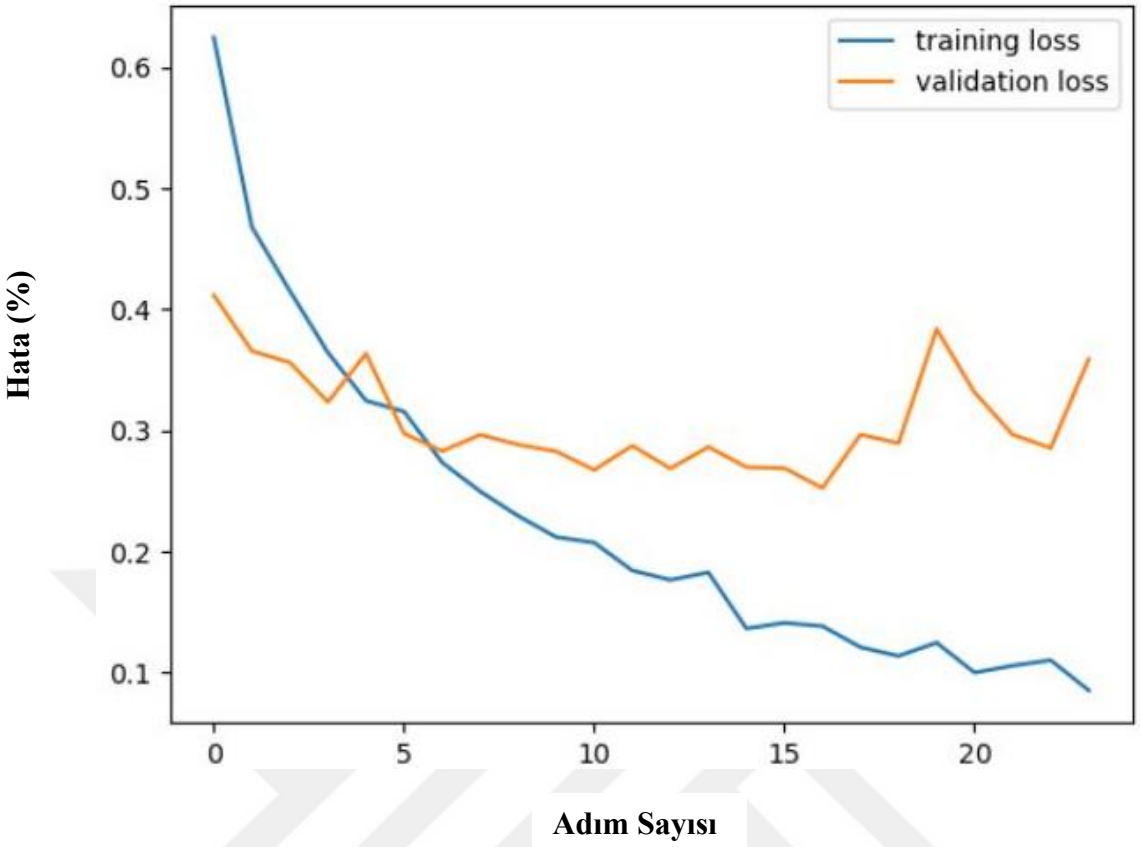
Şekil 69. MobileNetV2 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 70. MobileNetV2 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 70’teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 71. MobileNetV2 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 71'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

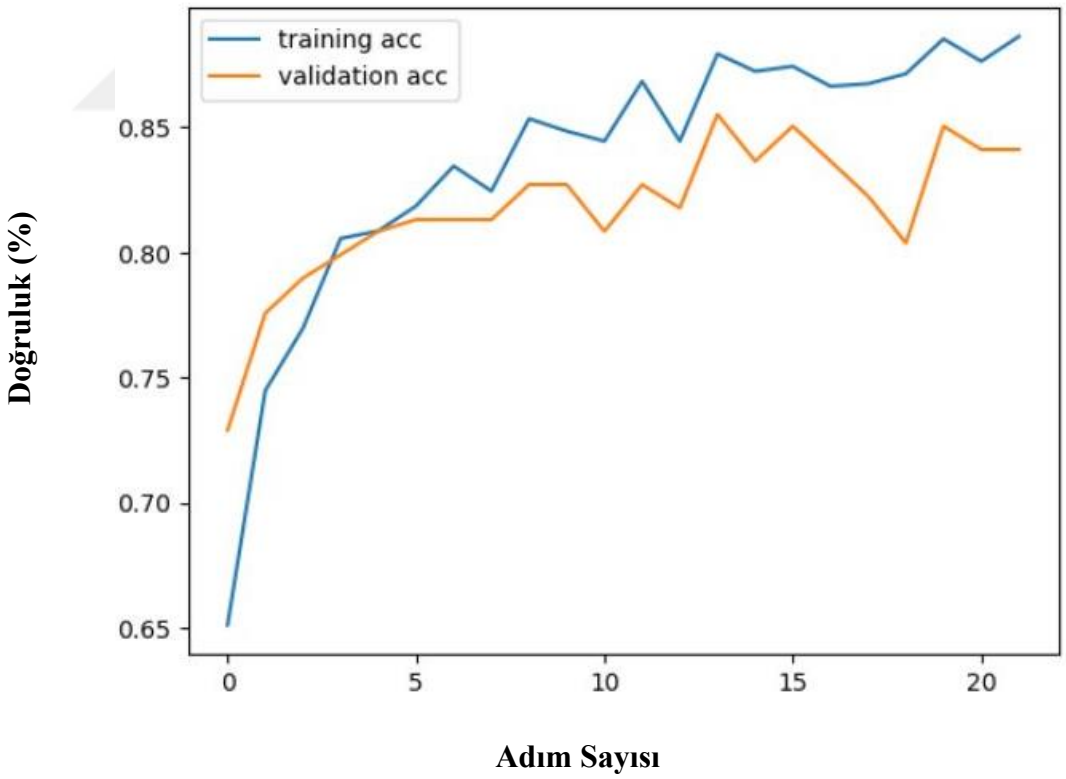
MobileNetV2 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen MobileNetV2 Tip – 2 modeli 22 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 16: acc did not improve from 0.87936
16/16 [==============================] - 32s 2s/step - loss: 0.2743 - acc: 0.8744 - val_loss: 0.2968 - val_acc: 0.8505
Epoch 17/60
16/16 [==============================] - ETA: 0s - loss: 0.2763 - acc: 0.8664
Epoch 17: acc did not improve from 0.87936
16/16 [==============================] - 32s 2s/step - loss: 0.2763 - acc: 0.8664 - val_loss: 0.2966 - val_acc: 0.8364
Epoch 18/60
16/16 [==============================] - ETA: 0s - loss: 0.2619 - acc: 0.8674
Epoch 18: acc did not improve from 0.87936
16/16 [==============================] - 32s 2s/step - loss: 0.2619 - acc: 0.8674 - val_loss: 0.3117 - val_acc: 0.8224
Epoch 19/60
16/16 [==============================] - ETA: 0s - loss: 0.2505 - acc: 0.8714
Epoch 19: acc did not improve from 0.87936
16/16 [==============================] - 33s 2s/step - loss: 0.2505 - acc: 0.8714 - val_loss: 0.3519 - val_acc: 0.8037
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.2362 - acc: 0.8853
Epoch 20: acc improved from 0.87936 to 0.88534, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 34s 2s/step - loss: 0.2362 - acc: 0.8853 - val_loss: 0.3287 - val_acc: 0.8505
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.2599 - acc: 0.8764
Epoch 21: acc did not improve from 0.88534
16/16 [==============================] - 33s 2s/step - loss: 0.2599 - acc: 0.8764 - val_loss: 0.3089 - val_acc: 0.8411
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.2353 - acc: 0.8863
Epoch 22: acc improved from 0.88534 to 0.88634, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 33s 2s/step - loss: 0.2353 - acc: 0.8863 - val_loss: 0.3382 - val_acc: 0.8411
```

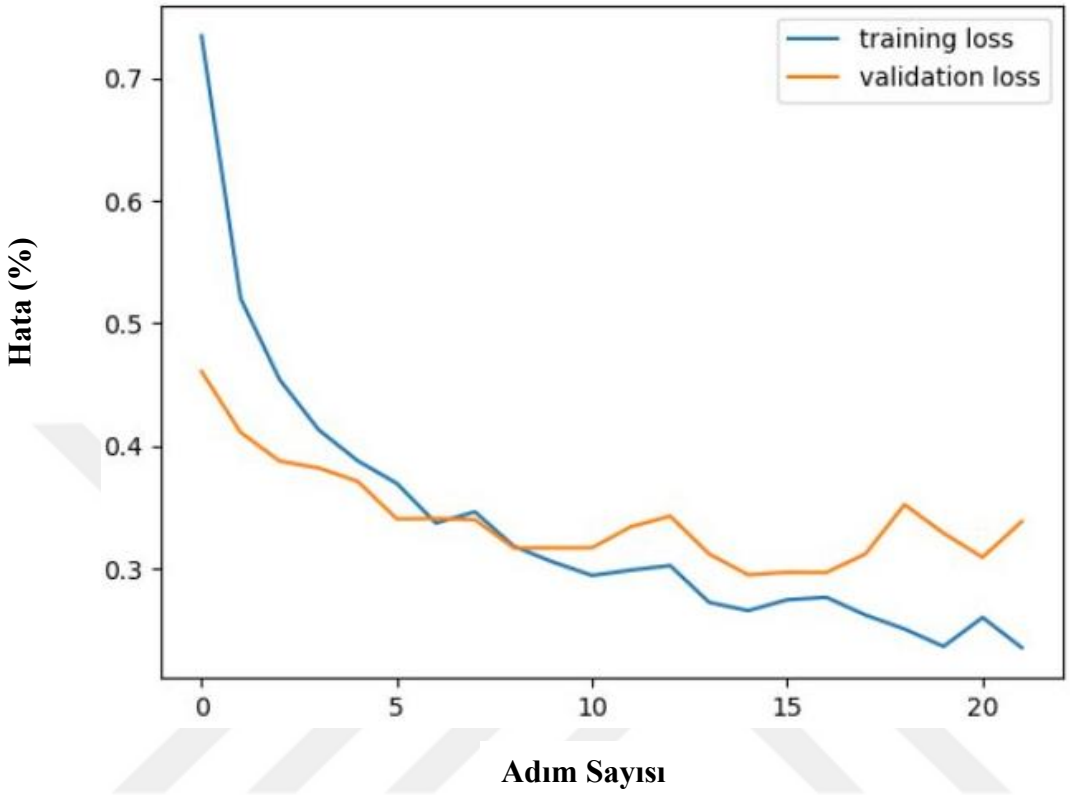
Şekil 72. MobileNetV2 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 73. MobileNetV2 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 73’teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 74. MobileNetV2 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 74’teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

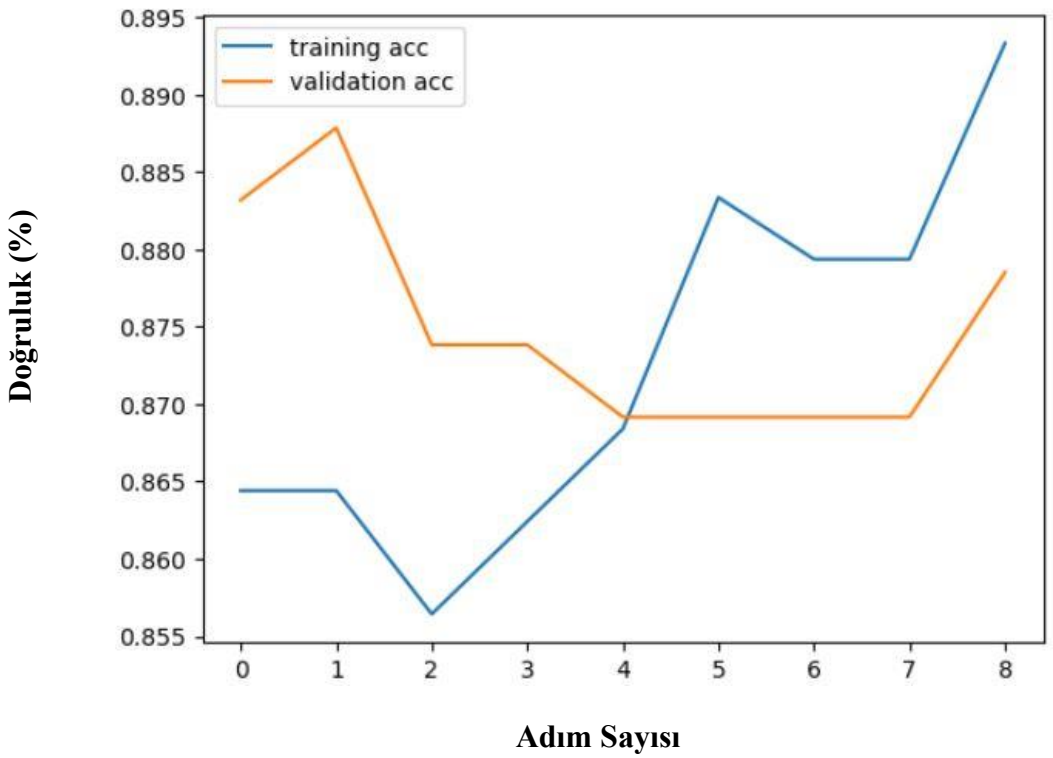
MobileNetV2 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen MobileNetV2 Tip – 3 modeli 9 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.2666 - acc: 0.8684
Epoch 5: acc improved from 0.86441 to 0.86839, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 35s 2s/step - loss: 0.2666 - acc: 0.8684 - val_loss: 0.2639 - val_acc: 0.8692
Epoch 6/60
16/16 [==============================] - ETA: 0s - loss: 0.2579 - acc: 0.8833
Epoch 6: acc improved from 0.86839 to 0.88335, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 34s 2s/step - loss: 0.2579 - acc: 0.8833 - val_loss: 0.2937 - val_acc: 0.8692
Epoch 7/60
16/16 [==============================] - ETA: 0s - loss: 0.2512 - acc: 0.8794
Epoch 7: acc did not improve from 0.88335
16/16 [==============================] - 34s 2s/step - loss: 0.2512 - acc: 0.8794 - val_loss: 0.2645 - val_acc: 0.8692
Epoch 8/60
16/16 [==============================] - ETA: 0s - loss: 0.2523 - acc: 0.8794
Epoch 8: acc did not improve from 0.88335
16/16 [==============================] - 33s 2s/step - loss: 0.2523 - acc: 0.8794 - val_loss: 0.2804 - val_acc: 0.8692
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.2281 - acc: 0.8933
Epoch 9: acc improved from 0.88335 to 0.89332, saving model to model\model_MobileNetV2_best.h5
16/16 [==============================] - 34s 2s/step - loss: 0.2281 - acc: 0.8933 - val_loss: 0.2860 - val_acc: 0.8785
```

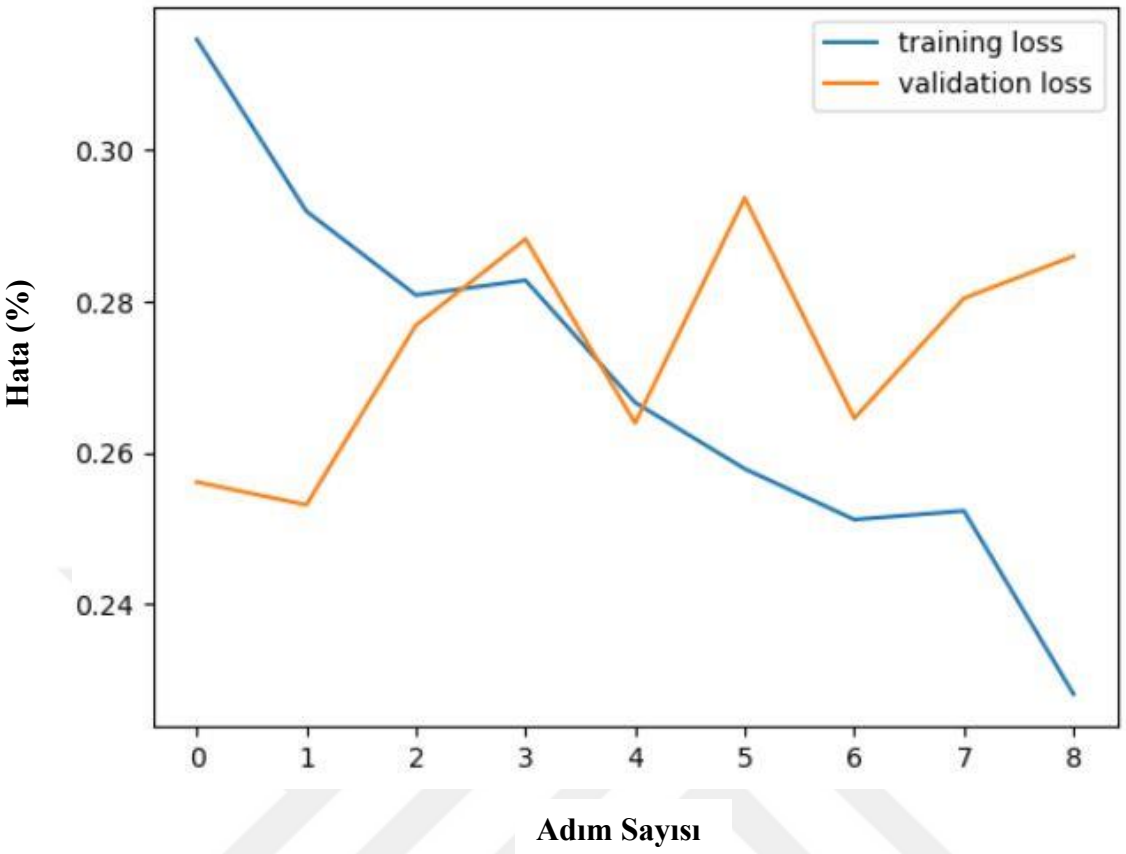
Şekil 75. MobileNetV2 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 76. MobileNetV2 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 76’daki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 77. MobileNetV2 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 77'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.5. InceptionResNetV2 Temelli Modelin Grafik ve Adım Sonuçları

InceptionResNetV2 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

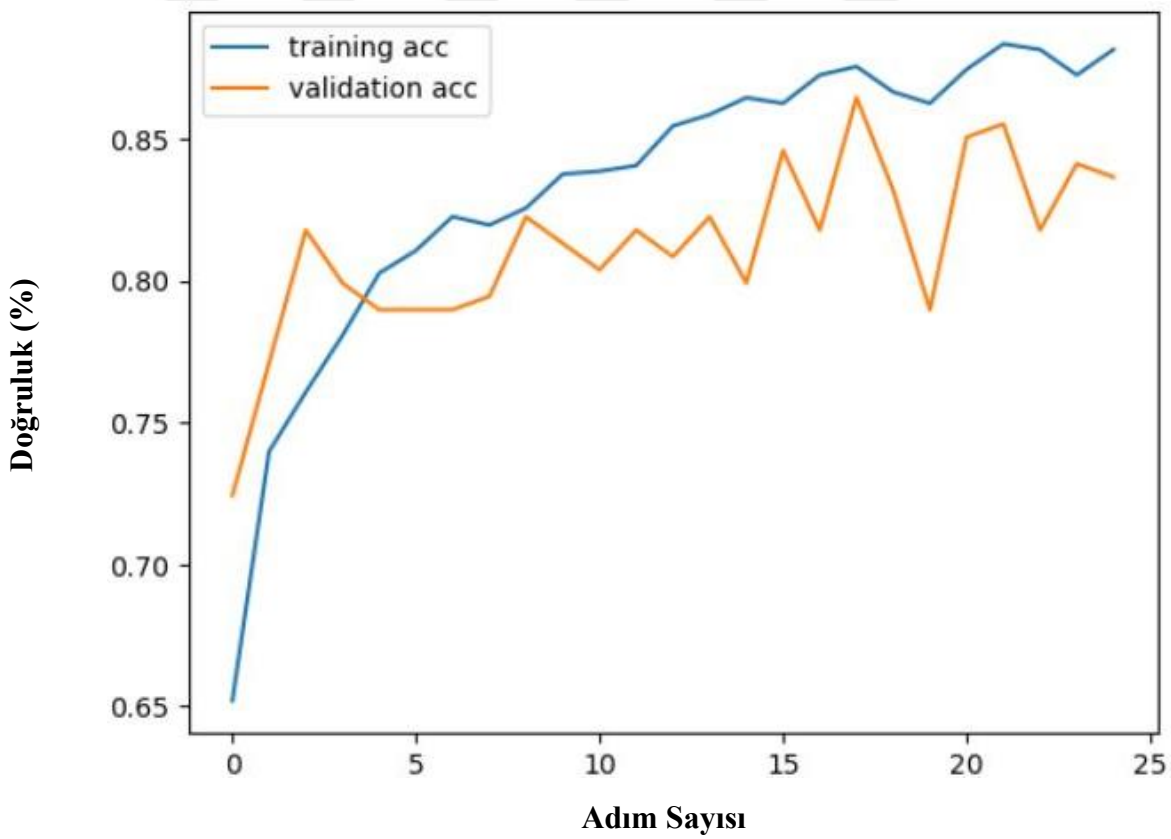
InceptionResNetV2 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionResNetV2 Tip – 1 modeli 25 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - ETA: 0s - loss: 0.2761 - acc: 0.8744
Epoch 21: acc did not improve from 0.87537
16/16 [==============================] - 97s 6s/step - loss: 0.2761 - acc: 0.8744 - val_loss: 0.3194 - val_acc: 0.8505
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.2684 - acc: 0.8833
Epoch 22: acc improved from 0.87537 to 0.88335, saving model to model\model_InceptionResNetV2_best.h5
16/16 [==============================] - 99s 6s/step - loss: 0.2684 - acc: 0.8833 - val_loss: 0.3387 - val_acc: 0.8551
Epoch 23/60
16/16 [==============================] - ETA: 0s - loss: 0.2722 - acc: 0.8814
Epoch 23: acc did not improve from 0.88335
16/16 [==============================] - 97s 6s/step - loss: 0.2722 - acc: 0.8814 - val_loss: 0.3440 - val_acc: 0.8178
Epoch 24/60
16/16 [==============================] - ETA: 0s - loss: 0.2647 - acc: 0.8724
Epoch 24: acc did not improve from 0.88335
16/16 [==============================] - 97s 6s/step - loss: 0.2647 - acc: 0.8724 - val_loss: 0.3281 - val_acc: 0.8411
Epoch 25/60
16/16 [==============================] - ETA: 0s - loss: 0.2598 - acc: 0.8814
Epoch 25: acc did not improve from 0.88335
16/16 [==============================] - 97s 6s/step - loss: 0.2598 - acc: 0.8814 - val_loss: 0.3310 - val_acc: 0.8364
```

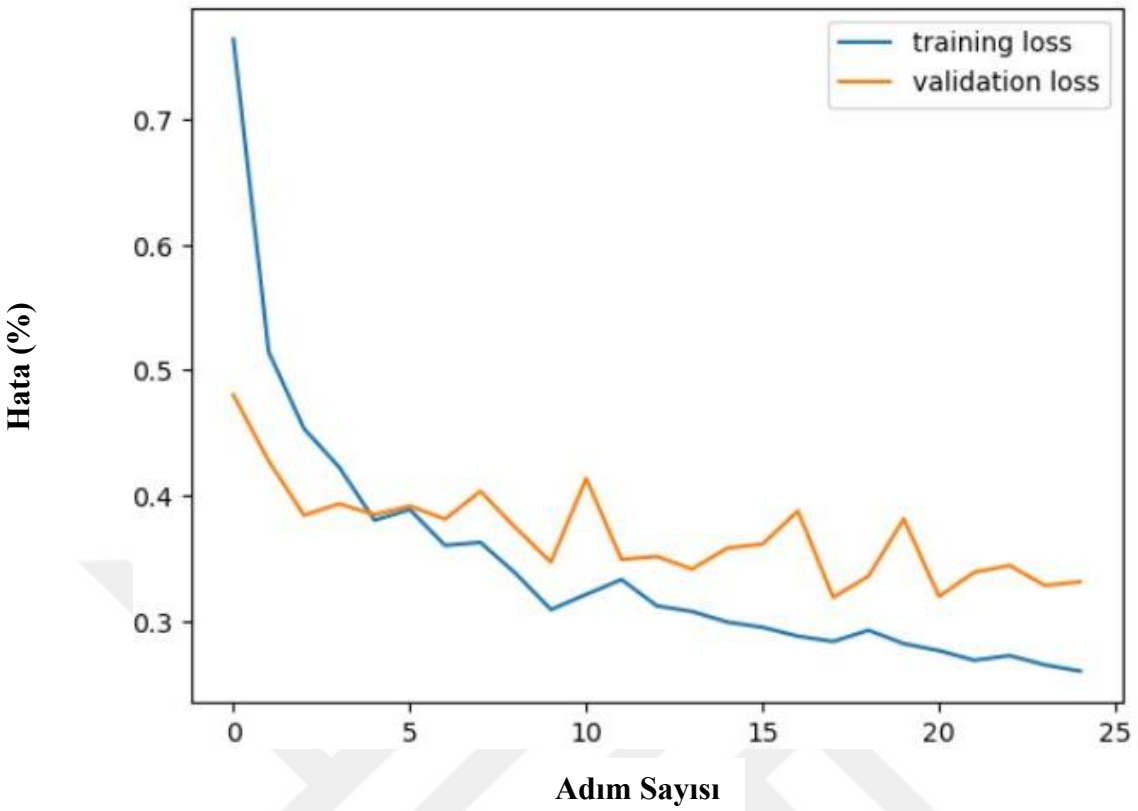
Şekil 78. InceptionResNetV2 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 79. InceptionResNetV2 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 79’daki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 80. InceptionResNetV2 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 80'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

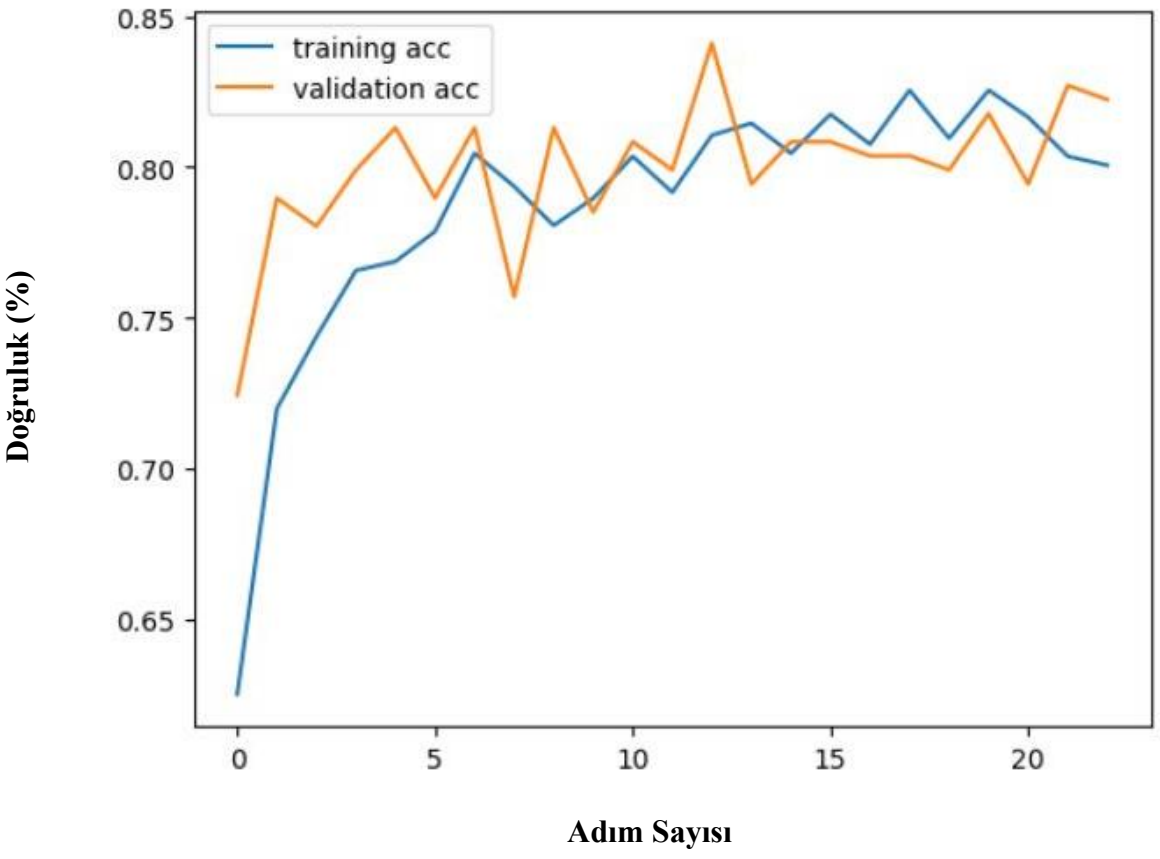
InceptionResNetV2 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionResNetV2 Tip – 2 modeli 23 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 19: acc did not improve from 0.82552
16/16 [==============================] - 95s 6s/step - loss: 0.3610 - acc: 0.8096 - val_loss: 0.3719 - val_acc: 0.7991
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.3628 - acc: 0.8255
Epoch 20: acc did not improve from 0.82552
16/16 [==============================] - 95s 6s/step - loss: 0.3628 - acc: 0.8255 - val_loss: 0.3662 - val_acc: 0.8178
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.3643 - acc: 0.8166
Epoch 21: acc did not improve from 0.82552
16/16 [==============================] - 95s 6s/step - loss: 0.3643 - acc: 0.8166 - val_loss: 0.3970 - val_acc: 0.7944
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.3839 - acc: 0.8036
Epoch 22: acc did not improve from 0.82552
16/16 [==============================] - 95s 6s/step - loss: 0.3839 - acc: 0.8036 - val_loss: 0.3632 - val_acc: 0.8271
Epoch 23/60
16/16 [==============================] - ETA: 0s - loss: 0.3811 - acc: 0.8006
Epoch 23: acc did not improve from 0.82552
16/16 [==============================] - 95s 6s/step - loss: 0.3811 - acc: 0.8006 - val_loss: 0.3702 - val_acc: 0.8224
```

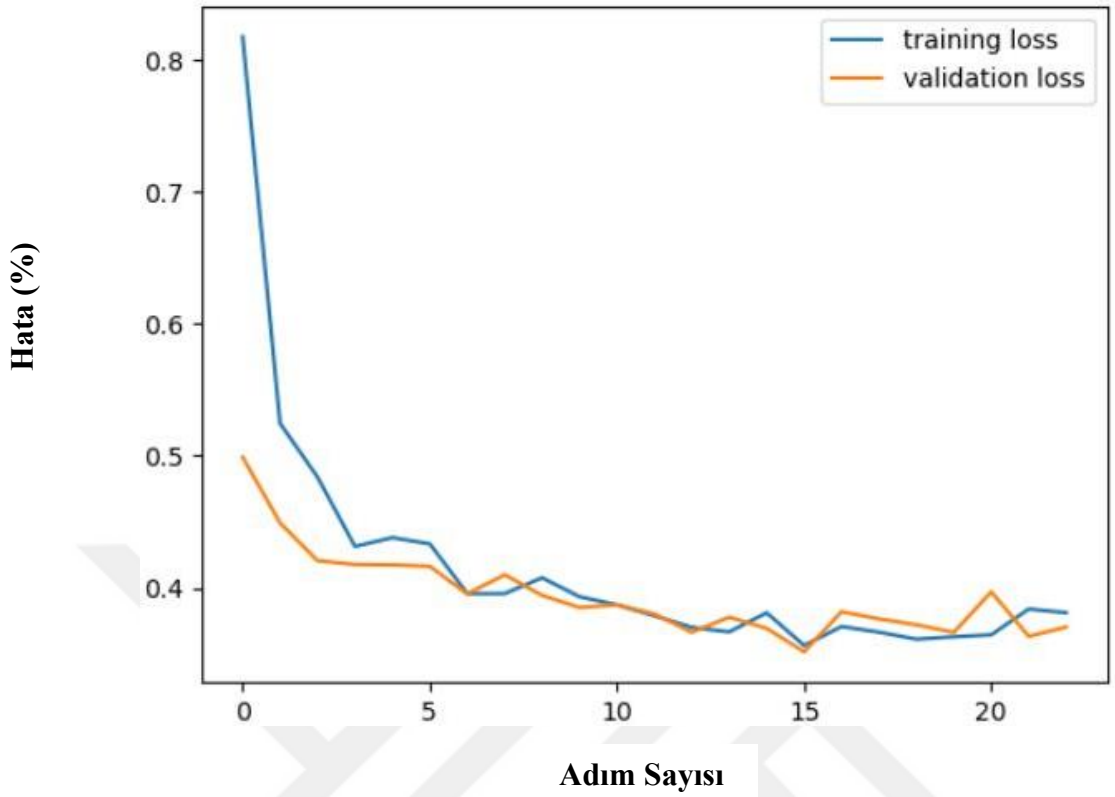
Şekil 81. InceptionResNetV2 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 82. InceptionResNetV2 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 82’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 83. InceptionResNetV2 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 83'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

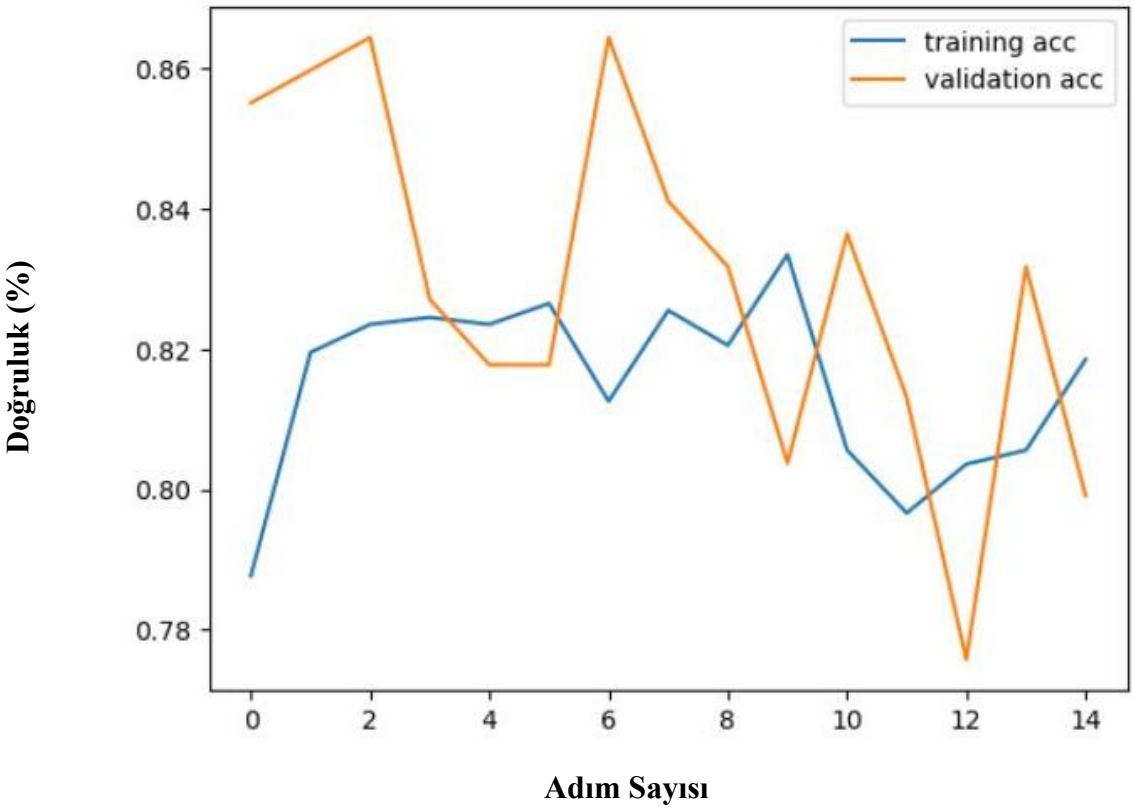
InceptionResNetV2 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionResNetV2 Tip – 3 modeli 15 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 12/60
16/16 [==============================] - ETA: 0s - loss: 0.3659 - acc: 0.7966
Epoch 12: acc did not improve from 0.83350
16/16 [==============================] - 100s 6s/step - loss: 0.3659 - acc: 0.7
966 - val_loss: 0.3561 - val_acc: 0.8131
Epoch 13/60
16/16 [==============================] - ETA: 0s - loss: 0.3670 - acc: 0.8036
Epoch 13: acc did not improve from 0.83350
16/16 [==============================] - 99s 6s/step - loss: 0.3670 - acc: 0.80
36 - val_loss: 0.3814 - val_acc: 0.7757
Epoch 14/60
16/16 [==============================] - ETA: 0s - loss: 0.3687 - acc: 0.8056
Epoch 14: acc did not improve from 0.83350
16/16 [==============================] - 100s 6s/step - loss: 0.3687 - acc: 0.8
056 - val_loss: 0.3353 - val_acc: 0.8318
Epoch 15/60
16/16 [==============================] - ETA: 0s - loss: 0.3537 - acc: 0.8185
Epoch 15: acc did not improve from 0.83350
16/16 [==============================] - 98s 6s/step - loss: 0.3537 - acc: 0.81
85 - val_loss: 0.3525 - val_acc: 0.7991
```

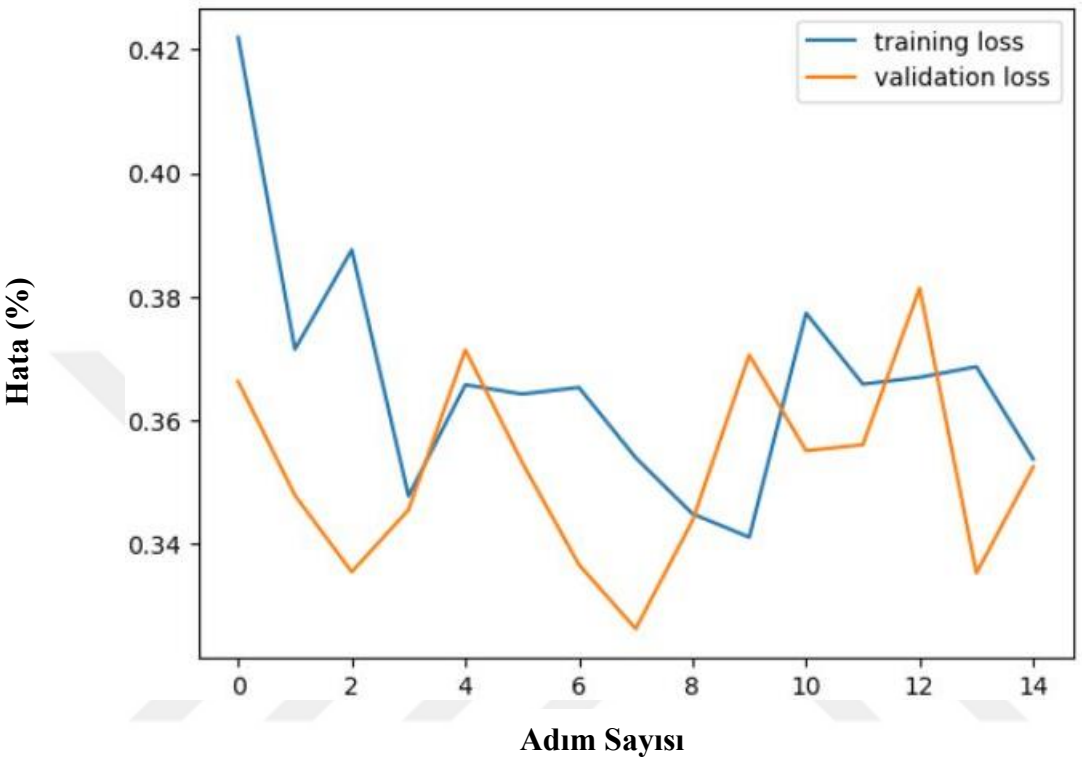
Şekil 84. InceptionResNetV2 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 85. InceptionResNetV2 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 85'teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 86. InceptionResNetV2 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 86'daki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.6. InceptionV3 Temelli Modelin Grafik ve Adım Sonuçları

InceptionV3 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

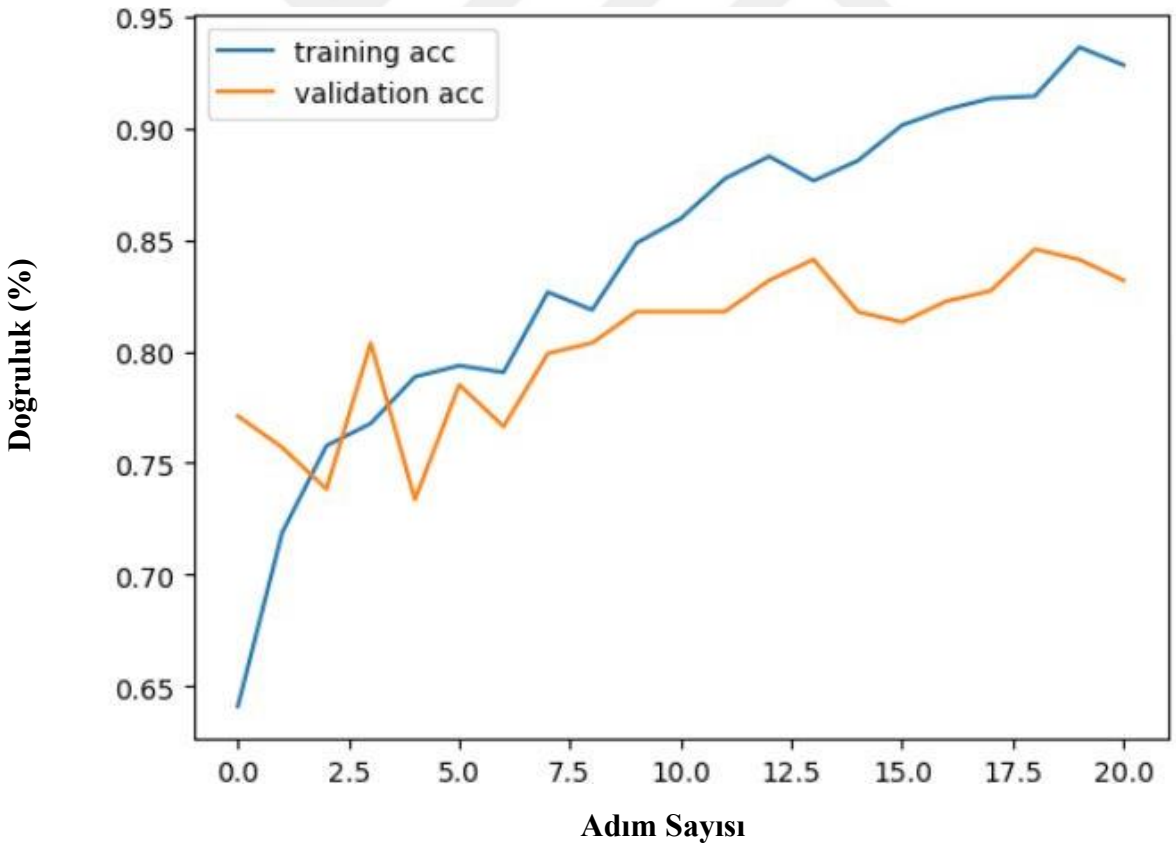
InceptionV3 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionV3 Tip – 1 modeli 21 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - 40s 3s/step - loss: 0.1975 - acc: 0.
9133 - val_loss: 0.4422 - val_acc: 0.8271
Epoch 19/60
16/16 [==============================] - ETA: 0s - loss: 0.1937 - acc: 0.9143
Epoch 19: acc improved from 0.91326 to 0.91426, saving model to model\model_I
nceptionResNetV2_best.h5
16/16 [==============================] - 41s 3s/step - loss: 0.1937 - acc: 0.
9143 - val_loss: 0.3943 - val_acc: 0.8458
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.1782 - acc: 0.9362
Epoch 20: acc improved from 0.91426 to 0.93619, saving model to model\model_I
nceptionResNetV2_best.h5
16/16 [==============================] - 40s 2s/step - loss: 0.1782 - acc: 0.
9362 - val_loss: 0.4015 - val_acc: 0.8411
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.1723 - acc: 0.9282
Epoch 21: acc did not improve from 0.93619
16/16 [==============================] - 40s 3s/step - loss: 0.1723 - acc: 0.
9282 - val_loss: 0.4679 - val_acc: 0.8318
```

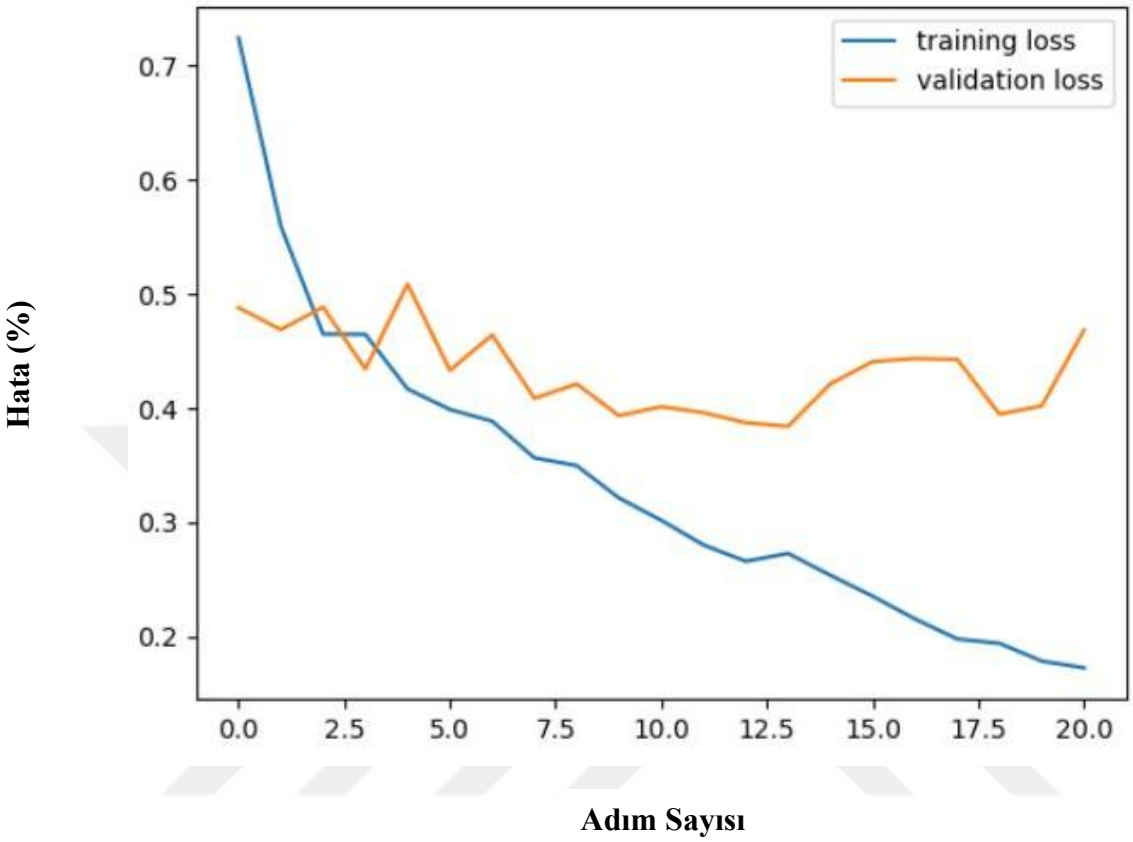
Şekil 87. InceptionV3 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 88. InceptionV3 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 88’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 89. InceptionV3 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 89’daki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

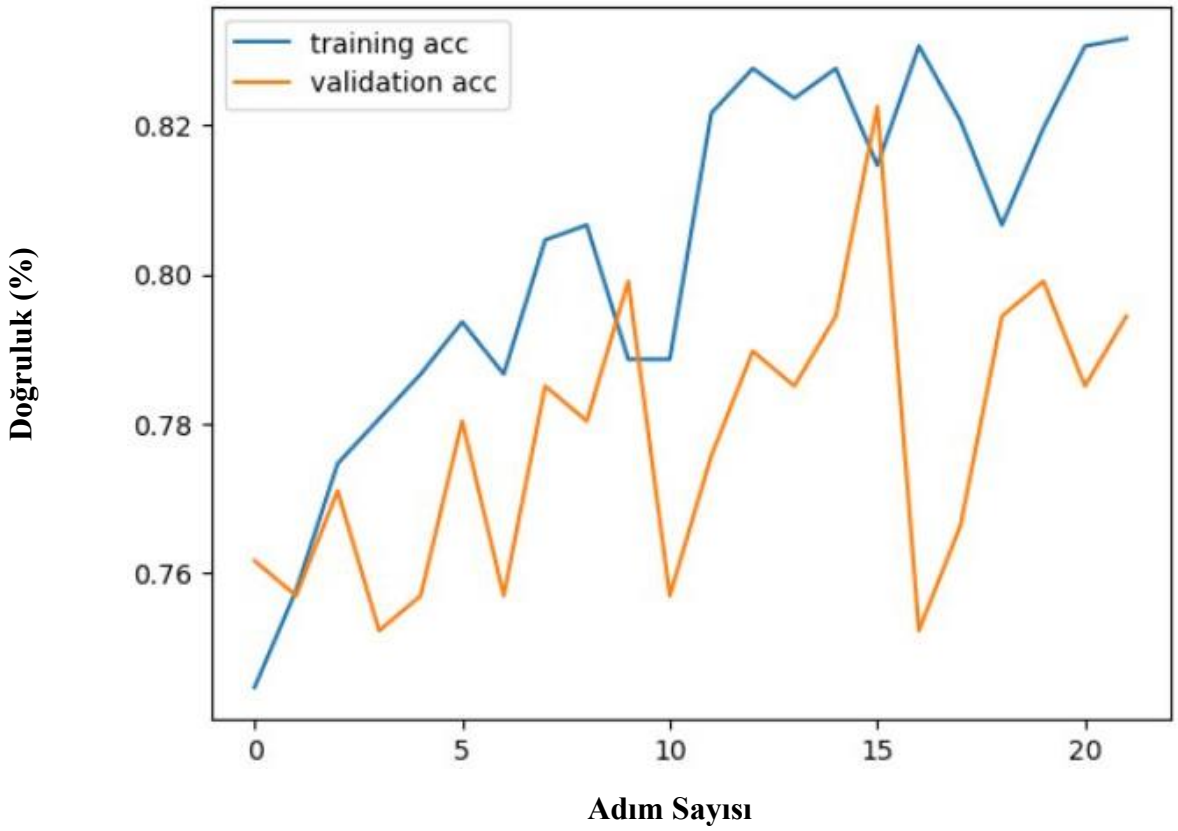
InceptionV3 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionV3 Tip – 2 modeli 22 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 19: acc did not improve from 0.83051
16/16 [==============================] - 42s 3s/step - loss: 0.3599 - acc: 0.
8066 - val_loss: 0.4146 - val_acc: 0.7944
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.3634 - acc: 0.8195
Epoch 20: acc did not improve from 0.83051
16/16 [==============================] - 42s 3s/step - loss: 0.3634 - acc: 0.
8195 - val_loss: 0.4447 - val_acc: 0.7991
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.3390 - acc: 0.8305
Epoch 21: acc did not improve from 0.83051
16/16 [==============================] - 43s 3s/step - loss: 0.3390 - acc: 0.
8305 - val_loss: 0.4341 - val_acc: 0.7850
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.3486 - acc: 0.8315
Epoch 22: acc improved from 0.83051 to 0.83151, saving model to model\model_I
nceptionv3_best.h5
16/16 [==============================] - 41s 3s/step - loss: 0.3486 - acc: 0.
8315 - val_loss: 0.4400 - val_acc: 0.7944
```

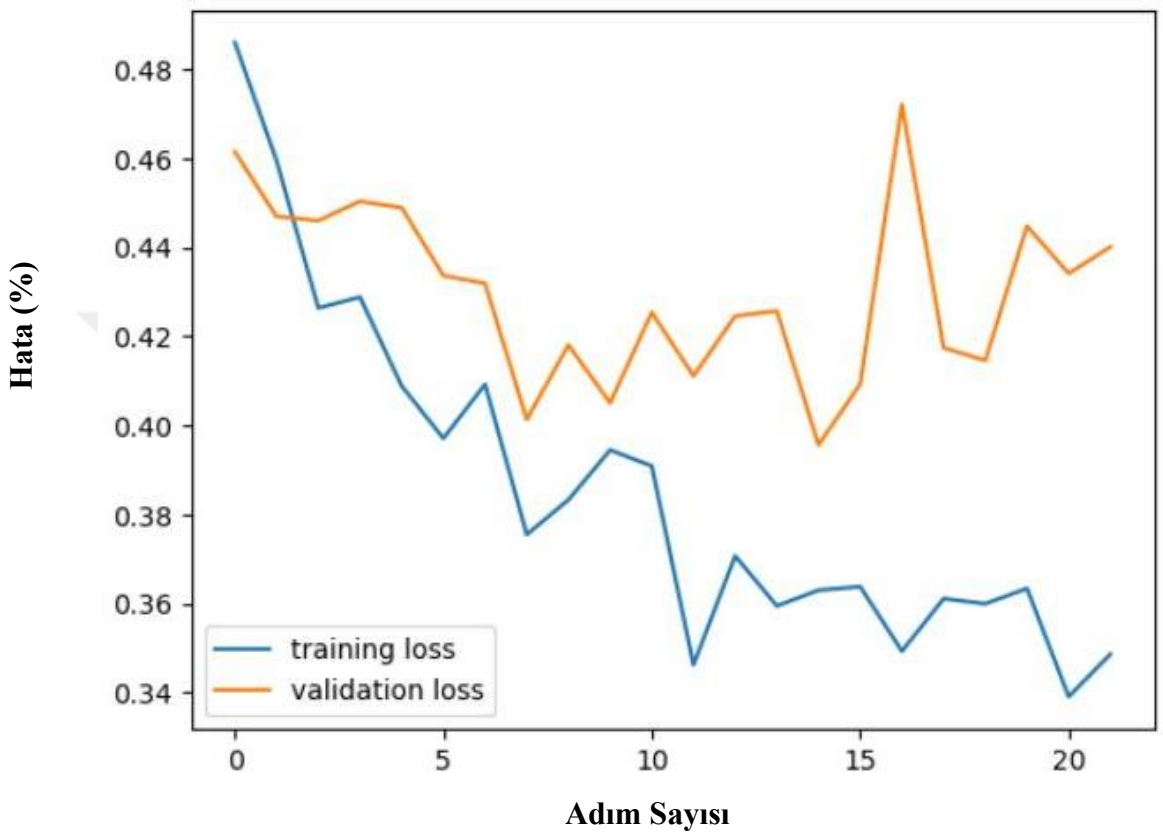
Şekil 90. InceptionV3 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 91. InceptionV3 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 91'deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir. Şekil 90'deki grafikte eğitim ve test verilerinin doğruluk skorları korelasyon oluşturmadığı gözlenmiştir.



Şekil 92. InceptionV3 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 92'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir. Şekil 91'deki grafikte modelin test hata skorlarının, eğitim hata skorlarından oldukça uzaklaştığı görülmektedir.

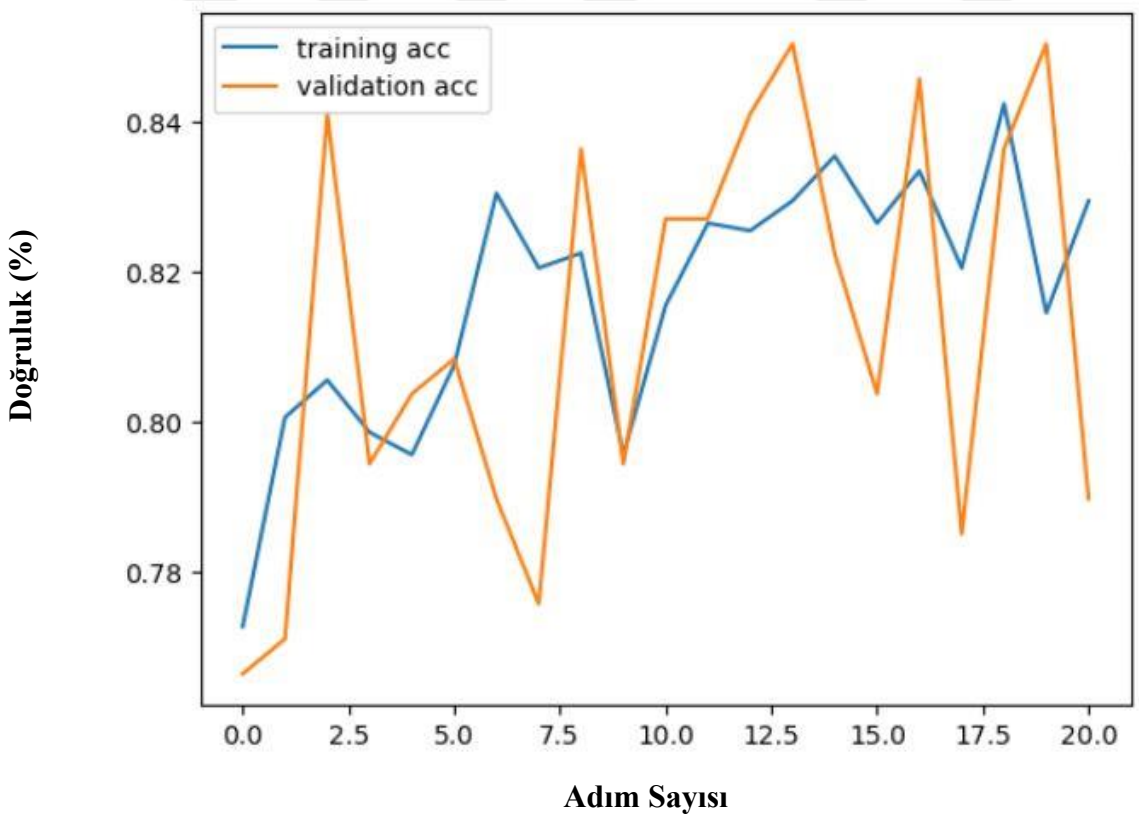
InceptionV3 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen InceptionV3 Tip – 3 modeli 21 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 18: acc did not improve from 0.83549
16/16 [==============================] - 43s 3s/step - loss: 0.3401 - acc: 0.
8205 - val_loss: 0.3973 - val_acc: 0.7850
Epoch 19/60
16/16 [==============================] - ETA: 0s - loss: 0.3126 - acc: 0.8425
Epoch 19: acc improved from 0.83549 to 0.84247, saving model to model\model_I
nceptionv3_best.h5
16/16 [==============================] - 43s 3s/step - loss: 0.3126 - acc: 0.
8425 - val_loss: 0.4081 - val_acc: 0.8364
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.3542 - acc: 0.8146
Epoch 20: acc did not improve from 0.84247
16/16 [==============================] - 45s 3s/step - loss: 0.3542 - acc: 0.
8146 - val_loss: 0.3685 - val_acc: 0.8505
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.3322 - acc: 0.8295
Epoch 21: acc did not improve from 0.84247
16/16 [==============================] - 43s 3s/step - loss: 0.3322 - acc: 0.
8295 - val_loss: 0.3683 - val_acc: 0.7897
```

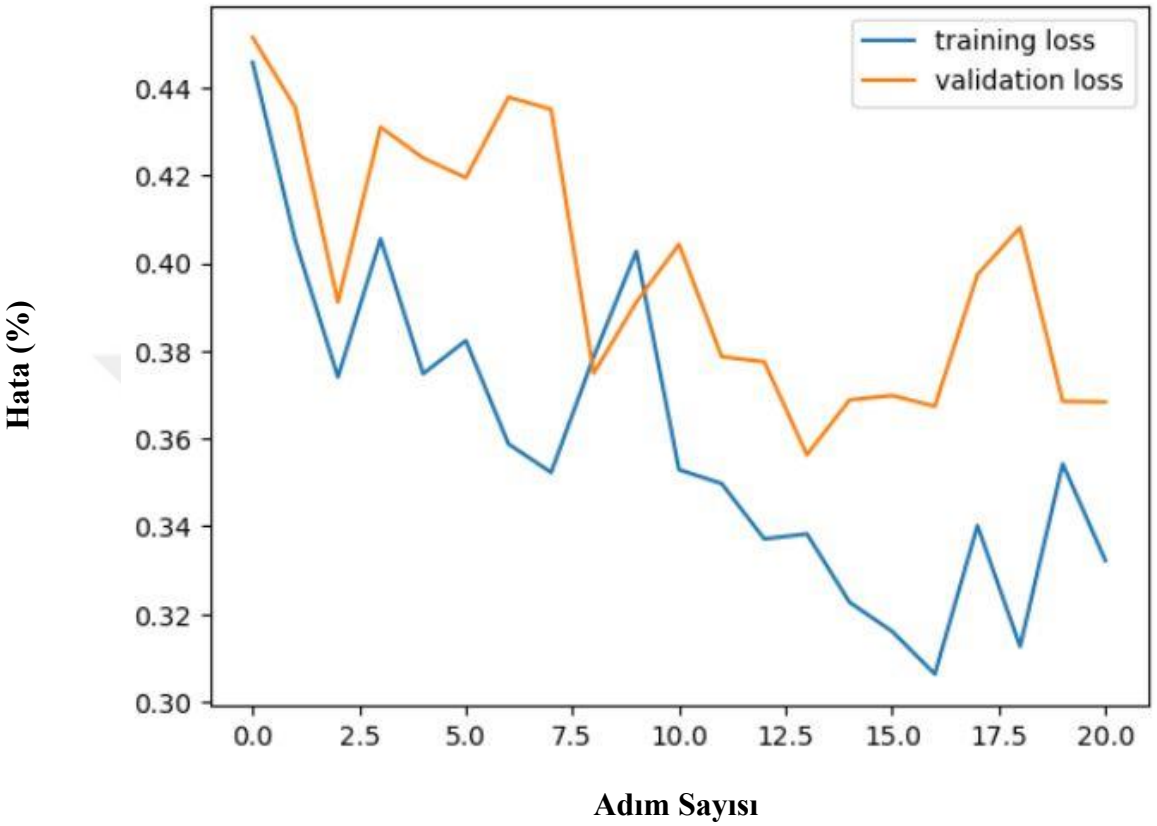
Şekil 93. InceptionV3 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 94. InceptionV3 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 94'teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 95. InceptionV3 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 95'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.7. ResNet50 Temelli Modelin Grafik ve Adım Sonuçları

ResNet50 bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

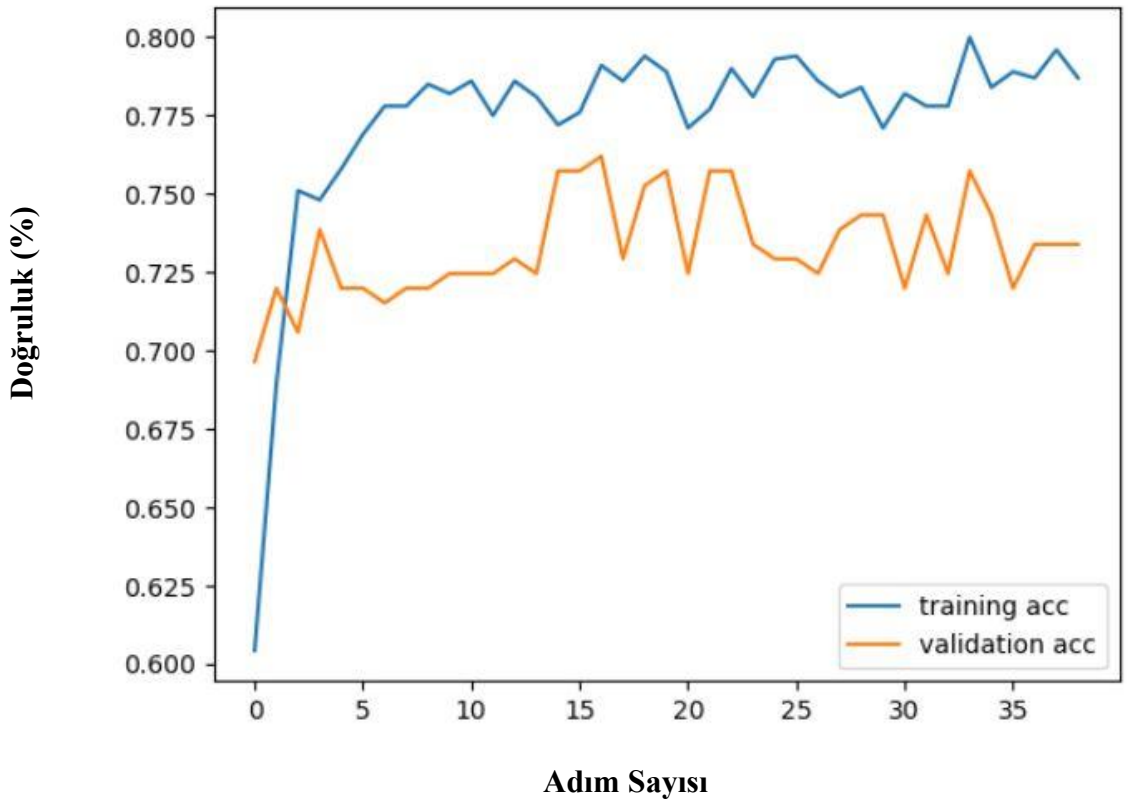
ResNet50 Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50 Tip – 1 modeli 39 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 36: acc did not improve from 0.79960
16/16 [==============================] - 84s 5s/step - loss: 0.4999 - acc: 0.
7886 - val_loss: 0.5289 - val_acc: 0.7196
Epoch 37/60
16/16 [==============================] - ETA: 0s - loss: 0.4999 - acc: 0.7866
Epoch 37: acc did not improve from 0.79960
16/16 [==============================] - 82s 5s/step - loss: 0.4999 - acc: 0.
7866 - val_loss: 0.5229 - val_acc: 0.7336
Epoch 38/60
16/16 [==============================] - ETA: 0s - loss: 0.4914 - acc: 0.7956
Epoch 38: acc did not improve from 0.79960
16/16 [==============================] - 82s 5s/step - loss: 0.4914 - acc: 0.
7956 - val_loss: 0.5209 - val_acc: 0.7336
Epoch 39/60
16/16 [==============================] - ETA: 0s - loss: 0.4809 - acc: 0.7866
Epoch 39: acc did not improve from 0.79960
16/16 [==============================] - 85s 5s/step - loss: 0.4809 - acc: 0.
7866 - val_loss: 0.5334 - val_acc: 0.7336
```

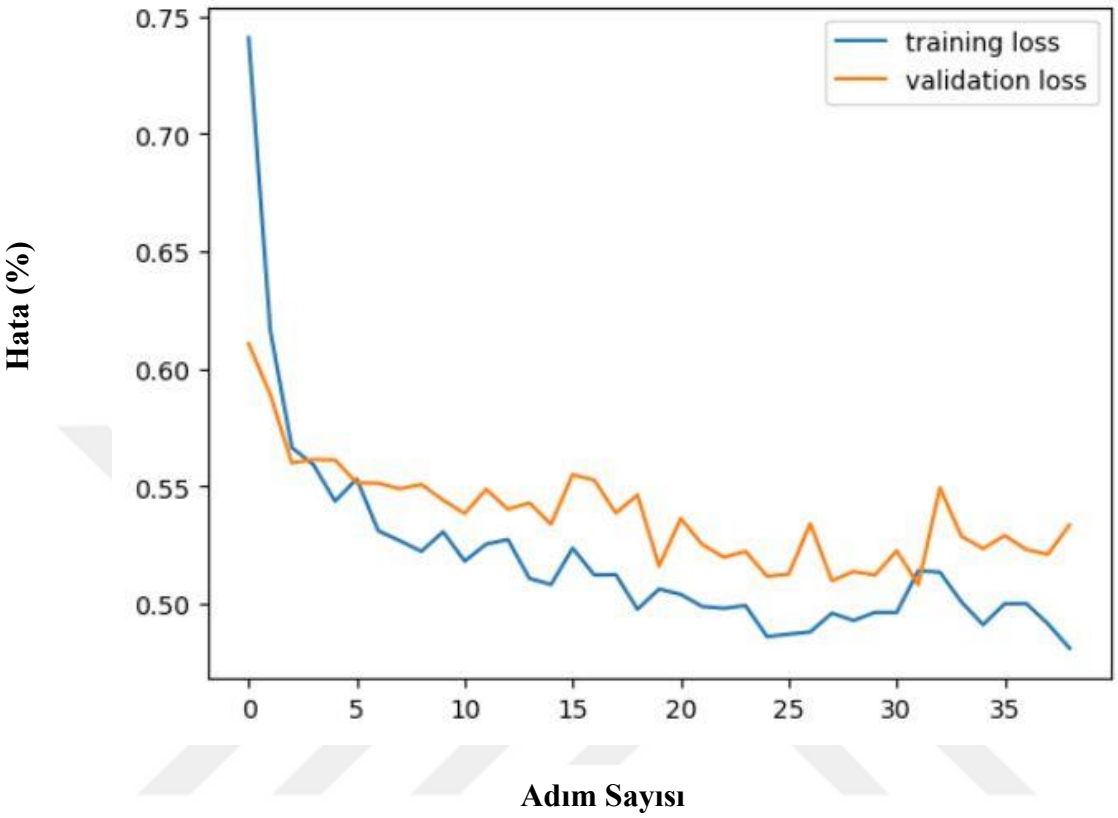
Şekil 96. ResNet50 Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 97. ResNet50 Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 97'deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 98. ResNet50 Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 98'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

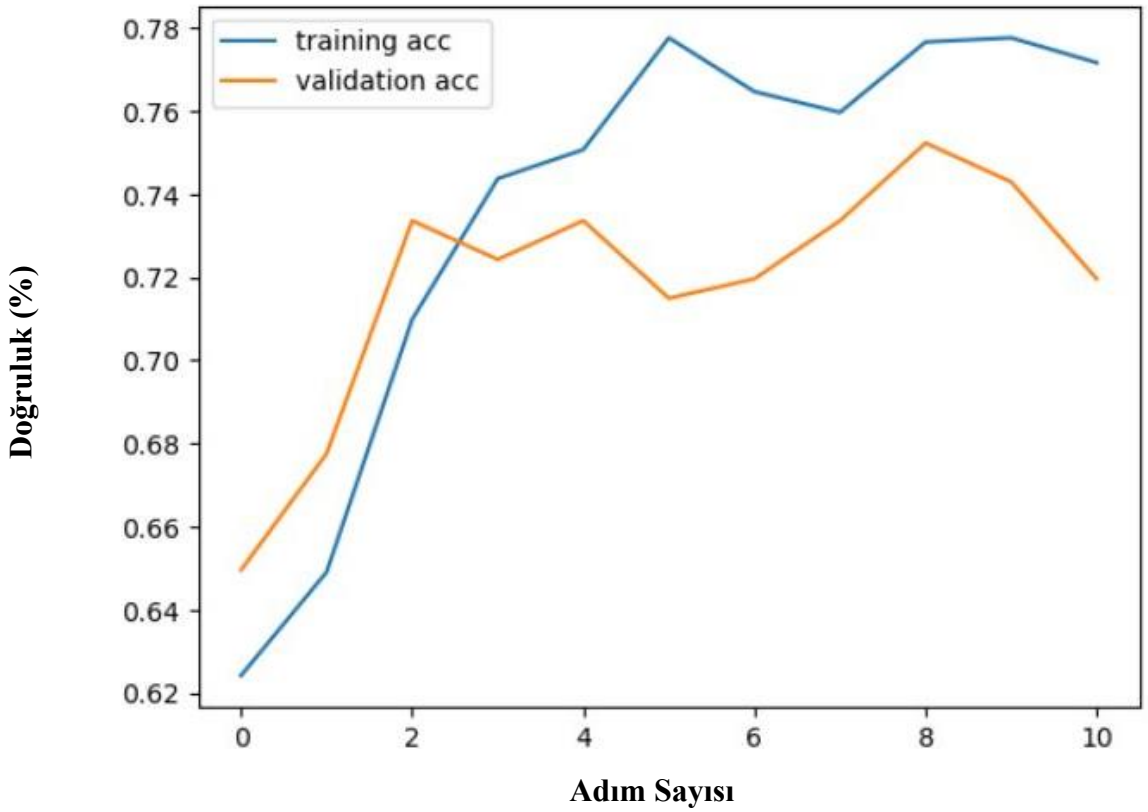
ResNet50 Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50 Tip – 2 modeli 11 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.5223 - acc: 0.7767
Epoch 9: acc did not improve from 0.77767
16/16 [==============================] - 79s 5s/step - loss: 0.5223 - acc: 0.77
67 - val_loss: 0.5822 - val_acc: 0.7523
Epoch 10/60
16/16 [==============================] - ETA: 0s - loss: 0.5163 - acc: 0.7777
Epoch 10: acc did not improve from 0.77767
16/16 [==============================] - 79s 5s/step - loss: 0.5163 - acc: 0.77
77 - val_loss: 0.5787 - val_acc: 0.7430
Epoch 11/60
16/16 [==============================] - ETA: 0s - loss: 0.5395 - acc: 0.7717
Epoch 11: acc did not improve from 0.77767
16/16 [==============================] - 79s 5s/step - loss: 0.5395 - acc: 0.77
17 - val_loss: 0.5586 - val_acc: 0.7196
```

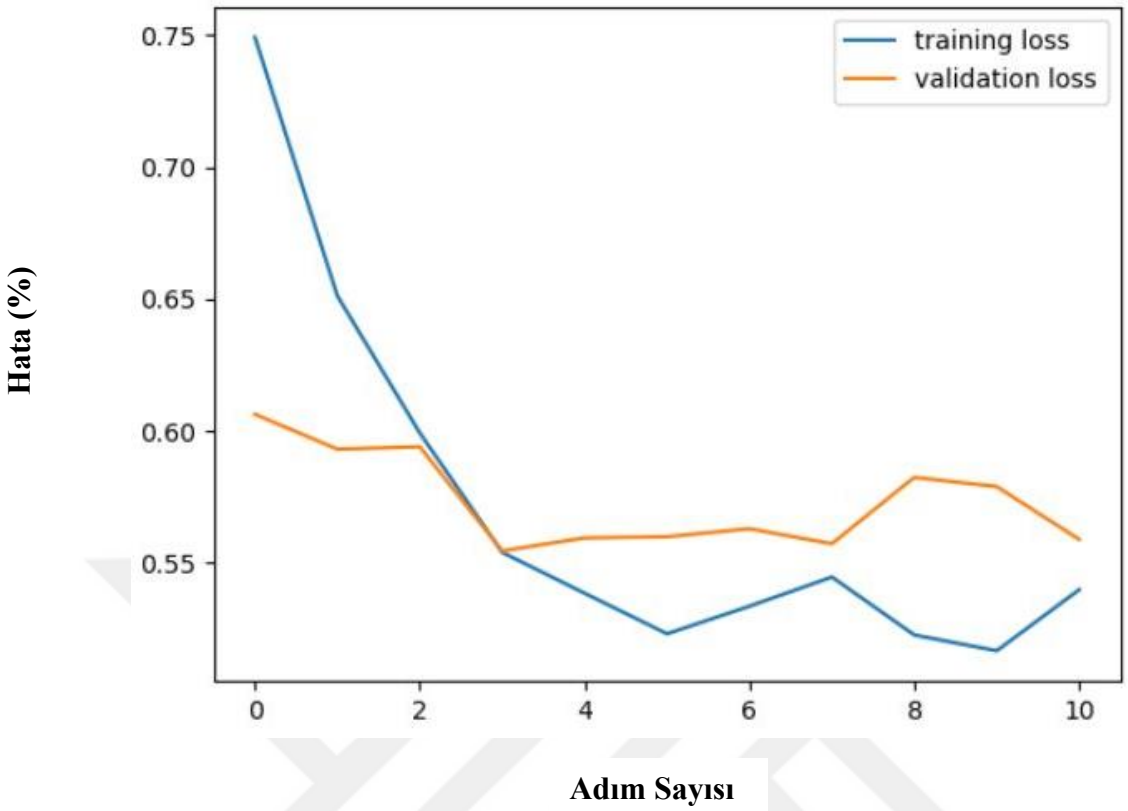
Şekil 99. ResNet50 Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 100. ResNet50 Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 100’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 101. ResNet50 Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 101'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

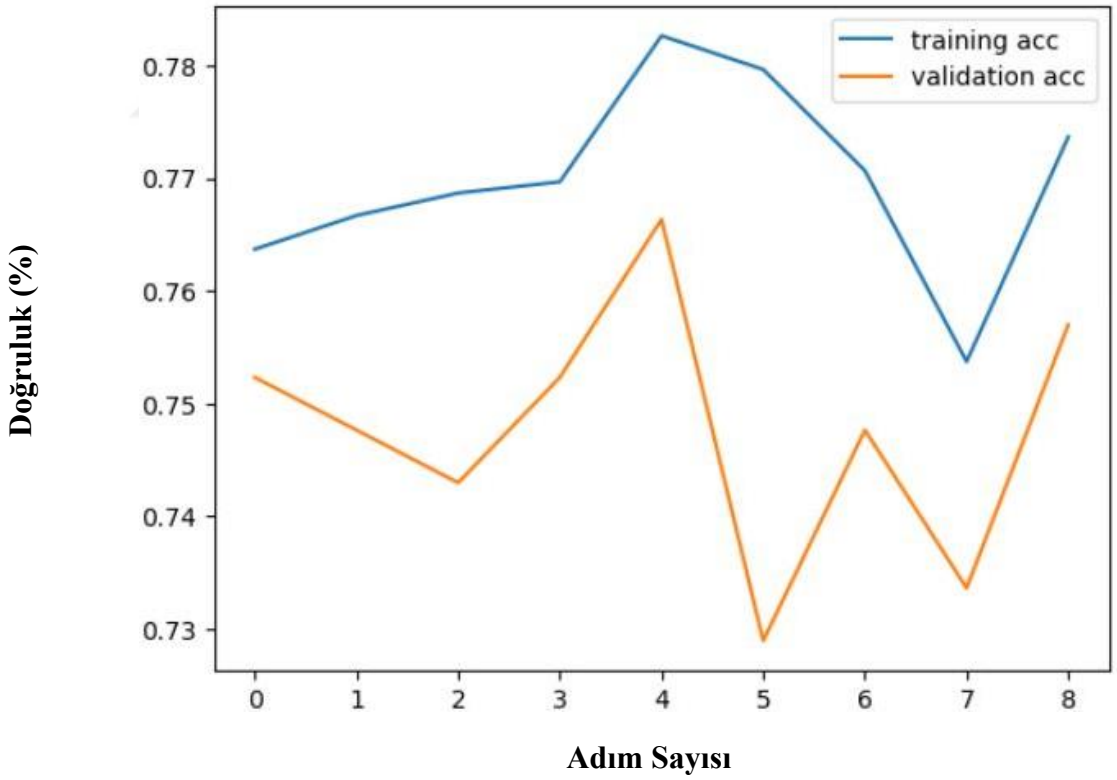
ResNet50 Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen ResNet50 Tip – 3 modeli 9 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 6/60
16/16 [==============================] - ETA: 0s - loss: 0.5072 - acc: 0.7797
Epoch 6: acc did not improve from 0.78265
16/16 [==============================] - 82s 5s/step - loss: 0.5072 - acc: 0.77
97 - val_loss: 0.5269 - val_acc: 0.7290
Epoch 7/60
16/16 [==============================] - ETA: 0s - loss: 0.5258 - acc: 0.7707
Epoch 7: acc did not improve from 0.78265
16/16 [==============================] - 82s 5s/step - loss: 0.5258 - acc: 0.77
07 - val_loss: 0.5169 - val_acc: 0.7477
Epoch 8/60
16/16 [==============================] - ETA: 0s - loss: 0.5419 - acc: 0.7537
Epoch 8: acc did not improve from 0.78265
16/16 [==============================] - 83s 5s/step - loss: 0.5419 - acc: 0.75
37 - val_loss: 0.5312 - val_acc: 0.7336
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.5130 - acc: 0.7737
Epoch 9: acc did not improve from 0.78265
16/16 [==============================] - 84s 5s/step - loss: 0.5130 - acc: 0.77
37 - val_loss: 0.5450 - val_acc: 0.7570
```

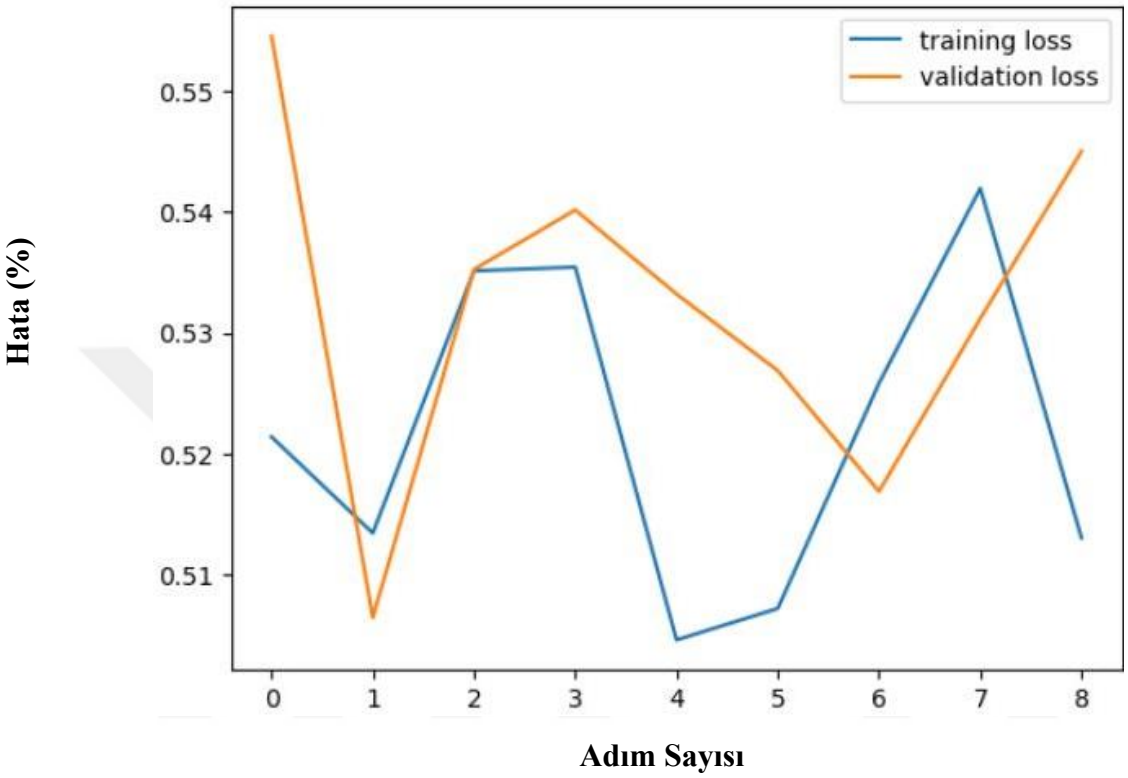
Şekil 102. ResNet50 Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 103. ResNet50 Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 103'teki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 104. ResNet50 Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 104'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

2.6.8. Xception Temelli Modelin Grafik ve Adım Sonuçları

Xception bazlı hazırlanan modeller 3 Tip şeklinde eğitilmiştir. Eğitim sonucunda oluşan grafikler ve adım gösterimleri detaylıca incelenmiştir.

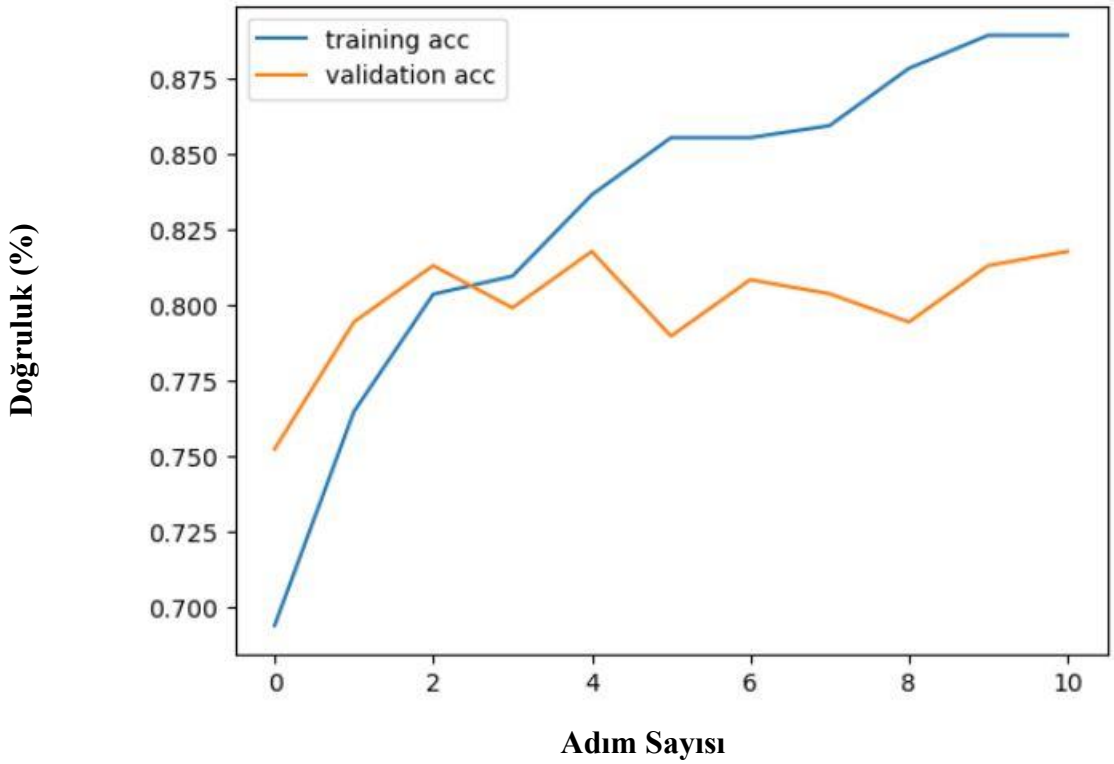
Xception Tip-1 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen Xception Tip – 1 modeli 11 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 9/60
16/16 [==============================] - ETA: 0s - loss: 0.2601 - acc: 0.8784
Epoch 9: acc improved from 0.85942 to 0.87836, saving model to model\model_vg
g16_best.h5
16/16 [==============================] - 77s 5s/step - loss: 0.2601 - acc: 0.
8784 - val_loss: 0.3839 - val_acc: 0.7944
Epoch 10/60
16/16 [==============================] - ETA: 0s - loss: 0.2303 - acc: 0.8893
Epoch 10: acc improved from 0.87836 to 0.88933, saving model to model\model_v
gg16_best.h5
16/16 [==============================] - 78s 5s/step - loss: 0.2303 - acc: 0.
8893 - val_loss: 0.4012 - val_acc: 0.8131
Epoch 11/60
16/16 [==============================] - ETA: 0s - loss: 0.2397 - acc: 0.8893
Epoch 11: acc did not improve from 0.88933
16/16 [==============================] - 77s 5s/step - loss: 0.2397 - acc: 0.
8893 - val_loss: 0.3965 - val_acc: 0.8178
```

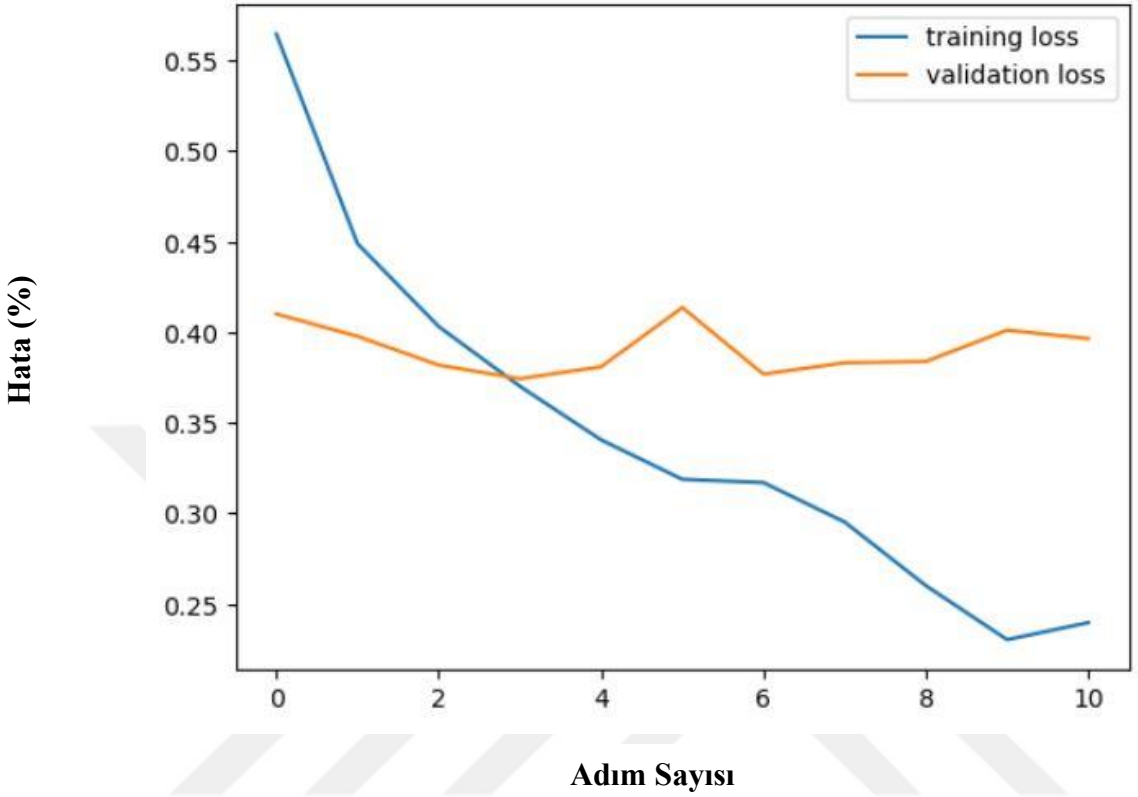
Şekil 105. Xception Tip-1 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 106. Xception Tip-1 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 106'daki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 107. Xception Tip-1 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 107'deki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

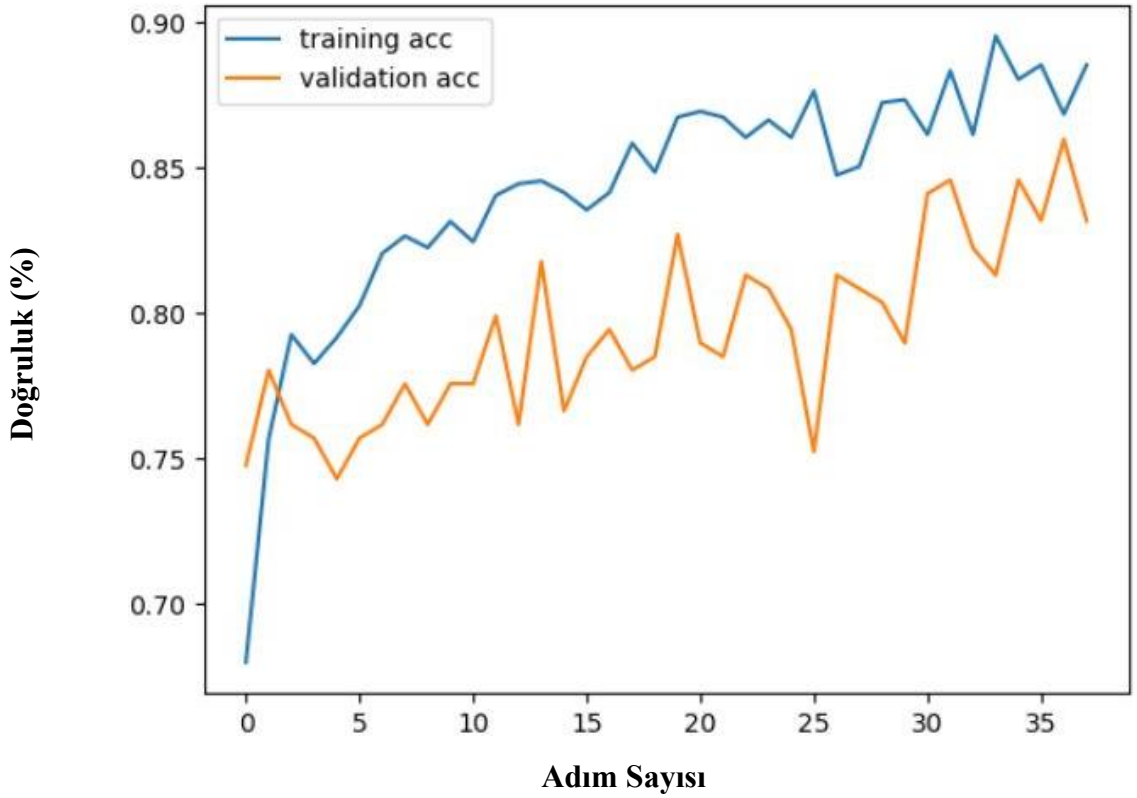
Xception Tip-2 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen Xception Tip – 2 modeli 38 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
16/16 [==============================] - 75s 5s/step - loss: 0.2346 - acc: 0.8953 - val_loss: 0.3469 - val_acc: 0.8131
Epoch 35/60
16/16 [==============================] - ETA: 0s - loss: 0.2669 - acc: 0.8804
Epoch 35: acc did not improve from 0.89531
16/16 [==============================] - 75s 5s/step - loss: 0.2669 - acc: 0.8804 - val_loss: 0.3319 - val_acc: 0.8458
Epoch 36/60
16/16 [==============================] - ETA: 0s - loss: 0.2727 - acc: 0.8853
Epoch 36: acc did not improve from 0.89531
16/16 [==============================] - 75s 5s/step - loss: 0.2727 - acc: 0.8853 - val_loss: 0.3659 - val_acc: 0.8318
Epoch 37/60
16/16 [==============================] - ETA: 0s - loss: 0.2863 - acc: 0.8684
Epoch 37: acc did not improve from 0.89531
16/16 [==============================] - 75s 5s/step - loss: 0.2863 - acc: 0.8684 - val_loss: 0.3330 - val_acc: 0.8598
Epoch 38/60
16/16 [==============================] - ETA: 0s - loss: 0.2437 - acc: 0.8853
Epoch 38: acc did not improve from 0.89531
16/16 [==============================] - 74s 5s/step - loss: 0.2437 - acc: 0.8853 - val_loss: 0.3446 - val_acc: 0.8318
```

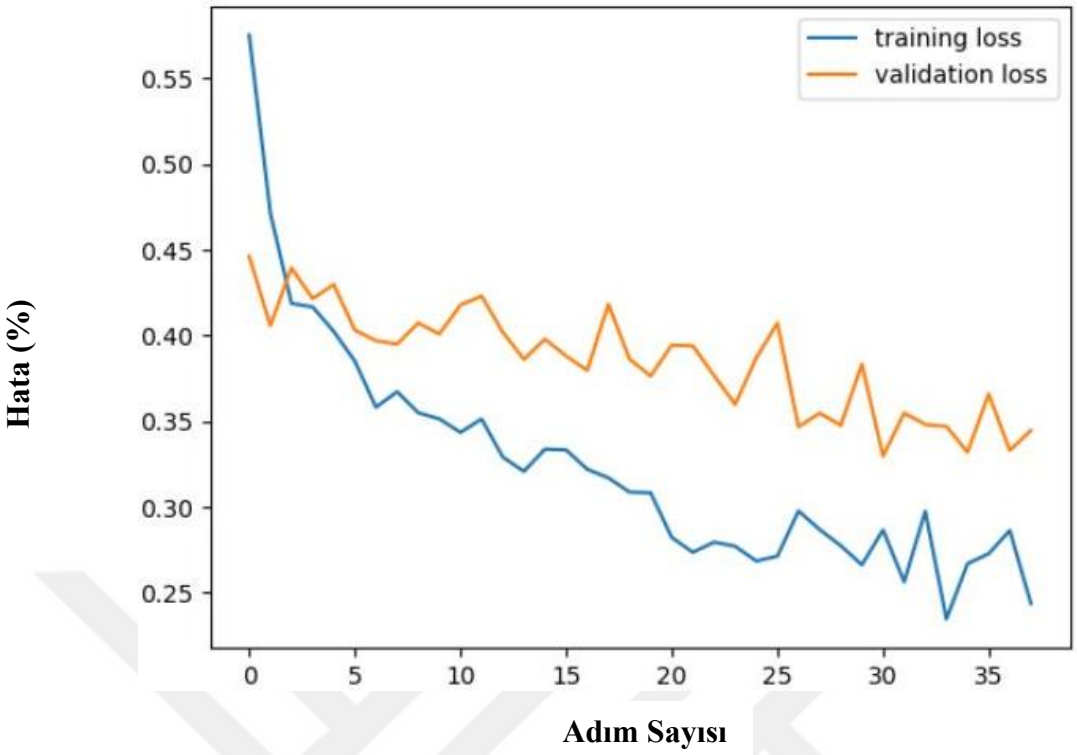
Şekil 108. Xception Tip-2 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 109. Xception Tip-2 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 109’daki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 110. Xception Tip-2 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 110'daki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

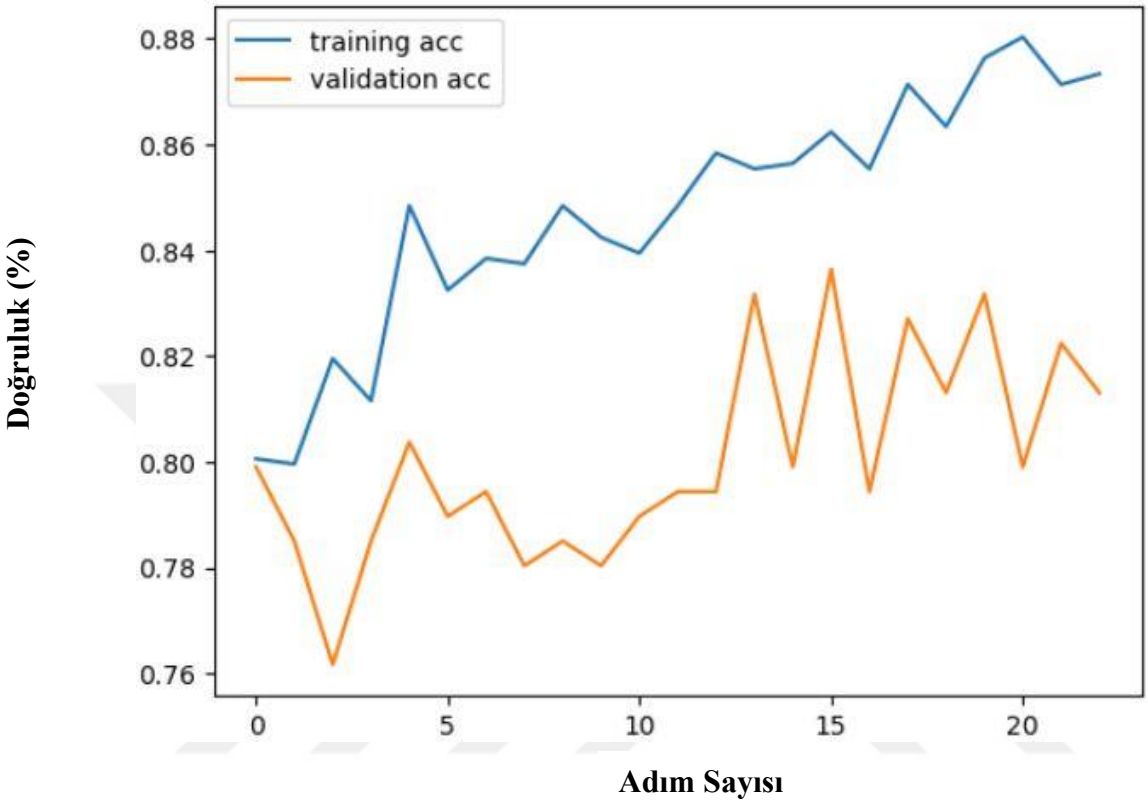
Xception Tip-3 Temelli Modelin Grafik ve Adım Sonuçları

Eğitilen Xception Tip – 3 modeli 23 adım sonra eğitimi aşırı uydurmaya maruz kalmamak için fonksiyon tarafından durdurulmuştur.

```
Epoch 20/60
16/16 [==============================] - ETA: 0s - loss: 0.2697 - acc: 0.8764
Epoch 20: acc improved from 0.87139 to 0.87637, saving model to model\model_vgg16_best.h5
16/16 [==============================] - 80s 5s/step - loss: 0.2697 - acc: 0.8764 - val_loss: 0.3472 - val_acc: 0.8318
Epoch 21/60
16/16 [==============================] - ETA: 0s - loss: 0.2825 - acc: 0.8804
Epoch 21: acc improved from 0.87637 to 0.88036, saving model to model\model_vgg16_best.h5
16/16 [==============================] - 81s 5s/step - loss: 0.2825 - acc: 0.8804 - val_loss: 0.3755 - val_acc: 0.7991
Epoch 22/60
16/16 [==============================] - ETA: 0s - loss: 0.2673 - acc: 0.8714
Epoch 22: acc did not improve from 0.88036
16/16 [==============================] - 80s 5s/step - loss: 0.2673 - acc: 0.8714 - val_loss: 0.3541 - val_acc: 0.8224
Epoch 23/60
16/16 [==============================] - ETA: 0s - loss: 0.2610 - acc: 0.8734
Epoch 23: acc did not improve from 0.88036
16/16 [==============================] - 80s 5s/step - loss: 0.2610 - acc: 0.8734 - val_loss: 0.3533 - val_acc: 0.8131
```

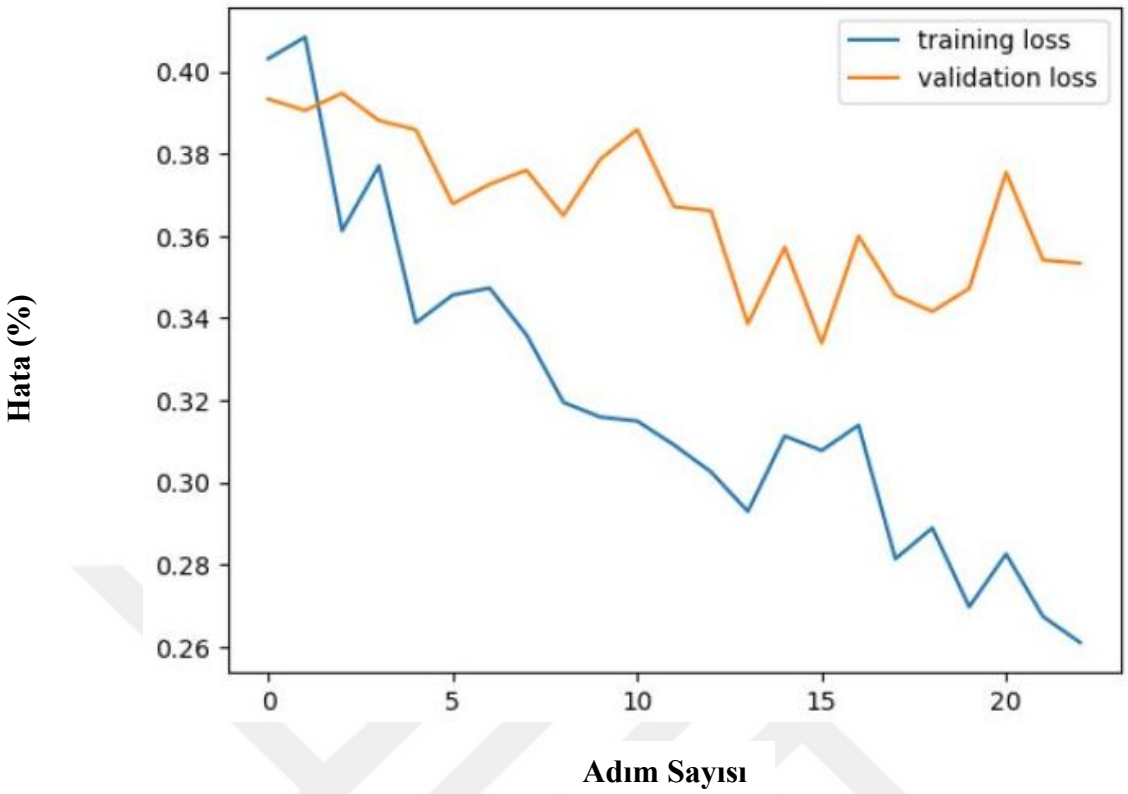
Şekil 111. Xception Tip-3 adım output görseli

Eğitim sonucunda eğitim ve doğrulama verilerinin oluşturduğu ‘doğruluk’ ve ‘hata’ grafiği elde edilmiştir.



Şekil 112. Xception Tip-3 eğitim-doğrulama verilerinin doğruluk-adım grafiği

Şekil 112’deki grafikte turuncu çizgi doğrulama verileri ile modelin doğruluk skorunu, mavi çizgi ise eğitim verileri ile modelin doğruluk skorunu göstermektedir.



Şekil 113. Xception Tip-3 eğitim-doğrulama verilerinin hata-adım grafiği

Şekil 113'teki grafikte turuncu çizgi doğrulama verileri ile modelin hata skorunu, mavi çizgi ise eğitim verileri ile modelin hata skorunu göstermektedir.

3. BULGULAR VE İRDELEME

X-ray bagaj tarama sistemlerinden elde edilen görüntülerin USB Bellek içeren ve içermeyen olarak sınıflandırılmasını amaçlayan bu tez çalışmasında Evrişim katmanını 8 farklı ön eğitimli model oluşturan 24 farklı model kullanılmıştır. Elde edilen metrik sonuçları detaylıca incelenmek üzere kaydedilmiştir.

3.1. Eğitim-Test Sonuçları ve Metrikler

Eğitim sonunda oluşan ağırlıkları belirlenmiş modellerin, sınıflandırma performansları belirlenmek için eğitimde kullanılan eğitim ve doğrulama verileri ile test edilmiştir. Daha sonra bütün oluşturulan modellere test verilerinden seçilen görüntüler sunulup metrik sonuçları kaydedilmiştir.

3.1.1. VGG16 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, VGG16 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

VGG16 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [9]: train_result = model_vgg16.evaluate(train_generator)
        val_result = model_vgg16.evaluate(test_generator)

        model_vgg16_result = pd.DataFrame(zip(train_result, val_result),
                                          columns=['Train', 'Validation'],
                                          index=['Loss', "Accuracy"])

        model_vgg16_result

        16/16 [==============================] - 125s 8s/step - loss: 0.2034 - acc: 0.9073
        4/4 [==============================] - 27s 6s/step - loss: 0.3229 - acc: 0.8505
```

Out[9]:

	Train	Validation
Loss	0.203432	0.322865
Accuracy	0.907278	0.850467

Şekil 114. VGG16 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [11]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         y_pred = model_vgg16.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 8s 4s/step
Confusion matrix:
 [[19  4]
 [ 3 38]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84        22
           1       0.93      0.90      0.92        42

    accuracy                           0.89        64
   macro avg       0.88      0.88      0.88        64
weighted avg       0.89      0.89      0.89        64
```

Şekil 115. VGG16 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,92 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

VGG16 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_vgg16.evaluate(train_generator)
        val_result = model_vgg16.evaluate(test_generator)

        model_vgg16_result = pd.DataFrame(zip(train_result, val_result),
                                          columns=['Train', 'Validation'],
                                          index=['Loss', "Accuracy"])

        model_vgg16_result
```

```
16/16 [==============================] - 126s 8s/step - loss: 0.2324 - acc: 0.8983
4/4 [==============================] - 27s 6s/step - loss: 0.3289 - acc: 0.8551
```

Out[7]:

	Train	Validation
Loss	0.232365	0.328853
Accuracy	0.898305	0.855140

Şekil 116. VGG16 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_vgg16.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 8s 4s/step
Confusion matrix:
 [[18  4]
 [ 3 39]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.82      0.86      0.84        21
           1       0.93      0.91      0.92        43

    accuracy                           0.89        64
   macro avg       0.87      0.88      0.88        64
weighted avg       0.89      0.89      0.89        64
```

Şekil 117. VGG16 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,92 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

VGG16 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
10]: train_result = model_vgg16_t.evaluate(train_generator)
     val_result = model_vgg16_t.evaluate(test_generator)

     model_vgg16_t_result = pd.DataFrame(zip(train_result, val_result),
                                         columns=['Train', 'Validation'],
                                         index=['Loss', "Accuracy"])

     model_vgg16_t_result
```

```
16/16 [==============================] - 127s 8s/step - loss: 0.2606 - acc: 0.8863
4/4 [==============================] - 29s 7s/step - loss: 0.3446 - acc: 0.8224
```

10]:

	Train	Validation
Loss	0.260578	0.344587

Şekil 118. VGG16 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [12]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_vgg16_t.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 8s 4s/step
Confusion matrix:
 [[19  5]
 [ 9 31]]
Accuracy Score:  0.78125
Classification report:
              precision    recall  f1-score   support

           0       0.79      0.68      0.73        28
           1       0.78      0.86      0.82        36

    accuracy                           0.78        64
   macro avg       0.78      0.77      0.77        64
weighted avg       0.78      0.78      0.78        64
```

Şekil 119. VGG16 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,78, F1-Skoru 0,82 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için Tip-1 ve Tip-2 modellere göre daha kötü bir sonuç vermiştir.

VGG16 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 1. VGG16 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
Vgg16 (tip-1)	0,20343	0,907278	0,322865	0,850467	0,92	0,890625
Vgg16 (tip-2)	0,23237	0,898305	0,328853	0,85514	0,92	0,890625
Vgg16 (tip-3)	0,26058	0,886341	0,344587	0,82243	0,82	0,78125

Tablo 1'de VGG16 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-1 ve Tip-2 modelin daha başarılı skorlar ürettiği görülmektedir.

3.1.2. VGG19 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, VGG19 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

VGG19 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_vgg19.evaluate(train_generator)
        val_result = model_vgg19.evaluate(test_generator)

        model_vgg19_result = pd.DataFrame(zip(train_result, val_result),
                                          columns=['Train', 'Validation'],
                                          index=['Loss', "Accuracy"])

        model_vgg19_result

        16/16 [==============================] - 169s 11s/step - loss: 0.1542 - acc: 0.9392
        4/4 [==============================] - 37s 9s/step - loss: 0.2616 - acc: 0.8738

Out[7]:
```

	Train	Validation
Loss	0.154191	0.261598
Accuracy	0.939182	0.873832

Şekil 120. VGG19 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_vgg19.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 11s 5s/step
Confusion matrix:
 [[18  6]
 [ 1 39]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.75      0.95      0.84        19
           1       0.97      0.87      0.92        45

    accuracy                           0.89        64
   macro avg       0.86      0.91      0.88        64
weighted avg       0.91      0.89      0.89        64
```

Şekil 121. VGG19 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,92 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

VGG19 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_vgg19.evaluate(train_generator)
        val_result = model_vgg19.evaluate(test_generator)

        model_vgg19_result = pd.DataFrame(zip(train_result, val_result),
                                          columns=['Train', 'Validation'],
                                          index=['Loss', "Accuracy"])

        model_vgg19_result
```

```
16/16 [==============================] - 169s 10s/step - loss: 0.2539 - acc: 0.8873
4/4 [==============================] - 35s 8s/step - loss: 0.3091 - acc: 0.8318
```

Out[7]:

	Train	Validation
Loss	0.253939	0.309147
Accuracy	0.887338	0.831776

Şekil 122. VGG19 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir. Ancak VGG19 Tip-1 skorlarına göre düşüş tespit edilmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_vgg19.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 11s 5s/step
Confusion matrix:
 [[22  6]
 [ 4 32]]
Accuracy Score:  0.84375
Classification report:
              precision    recall  f1-score   support

           0       0.79      0.85      0.81        26
           1       0.89      0.84      0.86        38

    accuracy                           0.84        64
   macro avg       0.84      0.84      0.84        64
weighted avg       0.85      0.84      0.84        64
```

Şekil 123. VGG19 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,84, F1-Skoru 0,86 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

VGG16 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_vgg19.evaluate(train_generator)
        val_result = model_vgg19.evaluate(test_generator)

        model_vgg19_result = pd.DataFrame(zip(train_result, val_result),
                                          columns=['Train', 'Validation'],
                                          index=['Loss', "Accuracy"])

        model_vgg19_result
```

```
16/16 [==============================] - 169s 11s/step - loss: 0.1942 - acc: 0.9322
4/4 [==============================] - 35s 8s/step - loss: 0.2620 - acc: 0.8785
```

Out[7]:

	Train	Validation
Loss	0.194196	0.262010
Accuracy	0.932203	0.878505

Şekil 124. VGG19 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_vgg19.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 10s 5s/step
Confusion matrix:
 [[19  4]
 [ 1 40]]
Accuracy Score:  0.921875
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.95      0.88        20
           1       0.98      0.91      0.94        44

    accuracy                           0.92        64
   macro avg       0.90      0.93      0.91        64
weighted avg       0.93      0.92      0.92        64
```

Şekil 125. VGG19 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,92, F1-Skoru 0,94 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için Tip-1 ve Tip-2 modellere göre daha başarılı bir sonuç vermiştir.

VGG19 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 2. VGG19 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	**loss**	**acc**	**val_loss**	**val_acc**	**f1-score**	**accuracy score**
Vgg19 (tip-1)	0.154191	0.939182	0.261598	0.873832	0.92	0.890625
Vgg19 (tip-2)	0.253939	0.887338	0.309147	0.831776	0.86	0.84375
Vgg19 (tip-3)	0.194196	0.932203	0.262010	0.878505	0.94	0.921875

Tablo 2'de VGG19 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-3 modelin daha başarılı skorlar ürettiği görülmektedir. Bu sayede, öğrenim aktarımı yönteminin başarılı bir sonuç verdiği yorumlanabilir.

3.1.3. ResNet50V2 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, ResNet50V2 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

ResNet50V2 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_ResNet50V2.evaluate(train_generator)
        val_result = model_ResNet50V2.evaluate(test_generator)

        model_ResNet50V2_result = pd.DataFrame(zip(train_result, val_result),
                                               columns=['Train', 'Validation'],
                                               index=['Loss', "Accuracy"])

        model_ResNet50V2_result

        16/16 [==============================] - 54s 3s/step - loss: 0.2361 - acc: 0.9163
        4/4 [==============================] - 12s 3s/step - loss: 0.3575 - acc: 0.8178
```

Out[8]:

	Train	Validation
Loss	0.236125	0.357501
Accuracy	0.916251	0.817757

Şekil 126. ResNet50V2 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_ResNet50V2.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 4s 2s/step
Confusion matrix:
 [[19  5]
 [ 3 37]]
Accuracy Score:  0.875
Classification report:
              precision    recall  f1-score   support

           0       0.79      0.86      0.83        22
           1       0.93      0.88      0.90        42

    accuracy                           0.88        64
   macro avg       0.86      0.87      0.86        64
weighted avg       0.88      0.88      0.88        64
```

Şekil 127. ResNet50V2 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,87, F1-Skoru 0,9 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

ResNet50V2 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_ResNet50V2.evaluate(train_generator)
val_result = model_ResNet50V2.evaluate(test_generator)

model_ResNet50V2_result = pd.DataFrame(zip(train_result, val_result),
                                       columns=['Train', 'Validation'],
                                       index=['Loss', "Accuracy"])

model_ResNet50V2_result
```

```
16/16 [==============================] - 52s 3s/step - loss: 0.1953 - acc: 0.9053
4/4 [==============================] - 11s 3s/step - loss: 0.2990 - acc: 0.8832
```

Out[7]:

	Train	Validation
Loss	0.195320	0.299025
Accuracy	0.905284	0.883178

Şekil 128. ResNet50V2 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir. ResNet50V2 Tip-1 skorlarına göre başarı kaydedilmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_ResNet50V2.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 4s 2s/step
Confusion matrix:
 [[17  4]
 [ 2 41]]
Accuracy Score:  0.90625
Classification report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        19
           1       0.95      0.91      0.93        45

    accuracy                           0.91        64
   macro avg       0.88      0.90      0.89        64
weighted avg       0.91      0.91      0.91        64
```

Şekil 129. ResNet50V2 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,93, F1-Skoru 0,9 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

ResNet50V2 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_ResNet50V2.evaluate(train_generator)
val_result = model_ResNet50V2.evaluate(test_generator)

model_ResNet50V2_result = pd.DataFrame(zip(train_result, val_result),
                                       columns=['Train', 'Validation'],
                                       index=['Loss', "Accuracy"])

model_ResNet50V2_result
```

```
16/16 [==============================] - 57s 4s/step - loss: 0.1927 - acc: 0.9123
4/4 [==============================] - 13s 3s/step - loss: 0.3101 - acc: 0.8738
```

Out[7]:

	Train	Validation
Loss	0.192718	0.310100
Accuracy	0.912263	0.873832

Şekil 130. ResNet50V2 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_ResNet50V2.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))

2/2 [==============================] - 5s 2s/step
Confusion matrix:
 [[17  4]
 [ 1 42]]
Accuracy Score:  0.921875
Classification report:
              precision    recall  f1-score   support

           0       0.81      0.94      0.87        18
           1       0.98      0.91      0.94        46

    accuracy                           0.92        64
   macro avg       0.89      0.93      0.91        64
weighted avg       0.93      0.92      0.92        64
```

Şekil 131. ResNet50V2 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,92, F1-Skoru 0,94 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için başarılı bulunmuştur.

ResNet50V2 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 3. ResNet50V2 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
ResNet50V2 (tip-1)	0.236125	0.916251	0.357501	0.817757	0.90	0.875
ResNet50V2 (tip-2)	0.195320	0.905284	0.299025	0.883178	0.90625	0.93
ResNet50V2 (tip-3)	0.192718	0.912263	0.310100	0.873832	0.94	0.921875

Tablo 3'te ResNet50V2 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-2 ve Tip-3 modelin daha başarılı skorlar ürettiği

görülmektedir. Bu sayede, veri arttırımı ve öğrenim aktarımı yöntemlerinin başarılı bir sonuç verdiği yorumlanabilir.

3.1.4. MobileNetV2 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, MobileNetV2 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

MobileNetV2 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_MobileNetV2.evaluate(train_generator)
        val_result = model_MobileNetV2.evaluate(test_generator)

        model_MobileNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                    columns=['Train', 'Validation'],
                                    index=['Loss', "Accuracy"])

        model_MobileNetV2_result

        16/16 [==============================] - 21s 1s/step - loss: 0.0710 - acc: 0.9791
        4/4 [==============================] - 5s 1s/step - loss: 0.2523 - acc: 0.8738
```

Out[8]:

	Train	Validation
Loss	0.071029	0.252285
Accuracy	0.979063	0.873832

Şekil 132. MobileNetV2 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         y_pred = model_MobileNetV2.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 2s 665ms/step
Confusion matrix:
 [[15  7]
 [ 0 42]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.68      1.00      0.81        15
           1       1.00      0.86      0.92        49

    accuracy                           0.89        64
   macro avg       0.84      0.93      0.87        64
weighted avg       0.93      0.89      0.90        64
```

Şekil 133. MobileNetV2 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,92 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

MobileNetV2 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_MobileNetV2.evaluate(train_generator)
        val_result = model_MobileNetV2.evaluate(test_generator)

        model_MobileNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                   columns=['Train', 'Validation'],
                                   index=['Loss', "Accuracy"])

        model_MobileNetV2_result
```

```
16/16 [==============================] - 22s 1s/step - loss: 0.2444 - acc: 0.8794
4/4 [==============================] - 5s 1s/step - loss: 0.2947 - acc: 0.8364
```

Out[7]:

	Train	Validation
Loss	0.244411	0.294663
Accuracy	0.879362	0.836449

Şekil 134. MobileNetV2 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir. Ancak MobileNetV2 Tip-1 skorlarına göre düşüş tespit edilmiştir.

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_MobileNetV2.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 2s 627ms/step
Confusion matrix:
 [[11  5]
 [ 6 42]]
Accuracy Score:  0.828125
Classification report:
              precision    recall  f1-score   support

           0       0.69      0.65      0.67        17
           1       0.88      0.89      0.88        47

    accuracy                           0.83        64
   macro avg       0.78      0.77      0.78        64
weighted avg       0.83      0.83      0.83        64
```

Şekil 135. MobileNetV2 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,82, F1-Skoru 0,88 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

MobileNetV2 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_MobileNetV2.evaluate(train_generator)
        val_result = model_MobileNetV2.evaluate(test_generator)

        model_MobileNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                    columns=['Train', 'Validation'],
                                    index=['Loss', "Accuracy"])

        model_MobileNetV2_result

        16/16 [==============================] - 22s 1s/step - loss: 0.1293 - acc: 0.9691
        4/4 [==============================] - 5s 1s/step - loss: 0.2531 - acc: 0.8879
```

Out[7]:

	Train	Validation
Loss	0.129258	0.253112
Accuracy	0.969093	0.887850

Şekil 136. MobileNetV2 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [11]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         y_pred = model_MobileNetV2.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))

         2/2 [==============================] - 1s 629ms/step
         Confusion matrix:
          [[13  6]
          [ 2 43]]
         Accuracy Score:  0.875
         Classification report:
                       precision    recall  f1-score   support

                    0       0.68      0.87      0.76        15
                    1       0.96      0.88      0.91        49

             accuracy                           0.88        64
            macro avg       0.82      0.87      0.84        64
         weighted avg       0.89      0.88      0.88        64
```

Şekil 137. MobileNetV2 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,87, F1-Skoru 0,91 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için Tip-2 modelinden başarılı olduğu tespit edilip ve Tip-1 modele göre başarısız olduğu gözlenmiştir.

MobileNetV2 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 4. MobileNetV2 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
MobileNetV2 (tip-1)	0.071029	0.979063	0.252285	0.873832	0.92	0.890625
MobileNetV2 (tip-2)	0.244411	0.879362	0.294663	0.836449	0.88	0.828125
MobileNetV2 (tip-3)	0.129258	0.969093	0.253112	0.887850	0.91	0.875

Tablo 4'te MobileNetV2 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-1 modelin daha başarılı skorlar ürettiği görülmektedir. Bu sayede, normal veriler ile eğitim gören modelin daha başarılı sonuç verdiği yorumlanabilir.

3.1.5. InceptionResNetV2 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, InceptionResNetV2 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

InceptionResNetV2 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_InceptionResNetV2.evaluate(train_generator)
        val_result = model_InceptionResNetV2.evaluate(test_generator)

        model_InceptionResNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                         columns=['Train', 'Validation'],
                                         index=['Loss', "Accuracy"])

        model_InceptionResNetV2_result

        16/16 [==============================] - 79s 5s/step - loss: 0.2193 - acc: 0.8993
        4/4 [==============================] - 17s 4s/step - loss: 0.3186 - acc: 0.8645
```

Out[8]:

	Train	Validation
Loss	0.219302	0.318633
Accuracy	0.899302	0.864486

Şekil 138. InceptionResNetV2 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         y_pred = model_InceptionResNetV2.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 8s 3s/step
Confusion matrix:
 [[16  2]
 [ 5 41]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.76      0.82        21
           1       0.89      0.95      0.92        43

    accuracy                           0.89        64
   macro avg       0.89      0.86      0.87        64
weighted avg       0.89      0.89      0.89        64
```

Şekil 139. InceptionResNetV2 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,92 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

3.1.5.2. InceptionResNetV2 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_InceptionResNetV2.evaluate(train_generator)
        val_result = model_InceptionResNetV2.evaluate(test_generator)

        model_InceptionResNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                  columns=['Train', 'Validation'],
                                  index=['Loss', "Accuracy"])

        model_InceptionResNetV2_result
```

```
16/16 [==============================] - 82s 5s/step - loss: 0.3186 - acc: 0.8315
4/4 [==============================] - 18s 4s/step - loss: 0.3514 - acc: 0.8084
```

Out[7]:

	Train	Validation
Loss	0.318589	0.351435
Accuracy	0.831505	0.808411

Şekil 140. InceptionResNetV2 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir. Ancak InceptionResNetV2 Tip-1 skorlarına göre düşüş tespit edilmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        y_pred = model_InceptionResNetV2.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 8s 3s/step
Confusion matrix:
 [[18 11]
 [ 1 34]]
Accuracy Score:  0.8125
Classification report:
              precision    recall  f1-score   support

           0       0.62      0.95      0.75        19
           1       0.97      0.76      0.85        45

    accuracy                           0.81        64
   macro avg       0.80      0.85      0.80        64
weighted avg       0.87      0.81      0.82        64
```

Şekil 141. InceptionResNetV2 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,81, F1-Skoru 0,85 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için Tip-1 modeline göre daha kötü sonuçlar vermiştir.

InceptionResNetV2 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_InceptionResNetV2.evaluate(train_generator)
        val_result = model_InceptionResNetV2.evaluate(test_generator)

        model_InceptionResNetV2_result = pd.DataFrame(zip(train_result, val_result),
                                    columns=['Train', 'Validation'],
                                    index=['Loss', "Accuracy"])

        model_InceptionResNetV2_result
```

```
16/16 [==============================] - 80s 5s/step - loss: 0.2541 - acc: 0.89
63
4/4 [==============================] - 17s 4s/step - loss: 0.3263 - acc: 0.8411
```

Out[8]:

	Train	Validation
Loss	0.254092	0.326297
Accuracy	0.896311	0.841121

Şekil 142. InceptionResNetV2 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
         y_pred = model_InceptionResNetV2.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))

2/2 [==============================] - 8s 3s/step
Confusion matrix:
 [[15  8]
 [ 4 37]]
Accuracy Score:  0.8125
Classification report:
              precision    recall  f1-score   support

           0       0.65      0.79      0.71        19
           1       0.90      0.82      0.86        45

    accuracy                           0.81        64
   macro avg       0.78      0.81      0.79        64
weighted avg       0.83      0.81      0.82        64
```

Şekil 143. InceptionResNetV2 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,86, F1-Skoru 0,81 olarak tespit edilmiştir.

InceptionResNetV2 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 5. InceptionResNetV2 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
InceptionResNetV2 (tip-1)	0.219302	0.899302	0.318633	0.864486	0.92	0.890625
InceptionResNetV2 (tip-2)	0.318589	0.831505	0.351435	0.808411	0.85	0.8125
InceptionResNetV2 (tip-3)	0.254092	0.896311	0.326297	0.841121	0.86	0.8125

Tablo 5'te InceptionResNetV2 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-1 modelin daha başarılı skorlar ürettiği görülmektedir. Bu sayede, normal veriler ile eğitim gören modelin daha başarılı sonuç verdiği yorumlanabilir.

3.1.6. InceptionV3 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, InceptionV3 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

InceptionV3 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_Inceptionv3.evaluate(train_generator)
        val_result = model_Inceptionv3.evaluate(test_generator)

        model_Inceptionv3_result = pd.DataFrame(zip(train_result, val_result),
                                    columns=['Train', 'Validation'],
                                    index=['Loss', "Accuracy"])

        model_Inceptionv3_result

        16/16 [==============================] - 31s 2s/step - loss: 0.2052 - acc: 0.92
        62
        4/4 [==============================] - 7s 2s/step - loss: 0.3836 - acc: 0.8411
```

Out[8]:

	Train	Validation
Loss	0.205166	0.383610
Accuracy	0.926221	0.841121

Şekil 144. InceptionV3 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
         y_pred = model_Inceptionv3.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 3s 1s/step
Confusion matrix:
 [[21  2]
 [ 7 34]]
Accuracy Score:  0.859375
Classification report:
              precision    recall  f1-score   support

           0       0.91      0.75      0.82        28
           1       0.83      0.94      0.88        36

    accuracy                           0.86        64
   macro avg       0.87      0.85      0.85        64
weighted avg       0.87      0.86      0.86        64
```

Şekil 145. InceptionV3 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,85, F1-Skoru 0,88 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermiştir.

InceptionV3 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_Inceptionv3.evaluate(train_generator)
        val_result = model_Inceptionv3.evaluate(test_generator)

        model_Inceptionv3_result = pd.DataFrame(zip(train_result, val_result),
                                   columns=['Train', 'Validation'],
                                   index=['Loss', "Accuracy"])

        model_Inceptionv3_result
```

```
16/16 [==============================] - 31s 2s/step - loss: 0.3380 - acc: 0.83
75
4/4 [==============================] - 7s 1s/step - loss: 0.3956 - acc: 0.7944
```

Out[8]:

	Train	Validation
Loss	0.338046	0.395603
Accuracy	0.837488	0.794393

Şekil 146. InceptionV3 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
         y_pred = model_Inceptionv3.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 3s 948ms/step
Confusion matrix:
 [[16  7]
 [ 5 36]]
Accuracy Score:  0.8125
Classification report:
              precision    recall  f1-score   support

           0       0.70      0.76      0.73        21
           1       0.88      0.84      0.86        43

    accuracy                           0.81        64
   macro avg       0.79      0.80      0.79        64
weighted avg       0.82      0.81      0.81        64
```

Şekil 147. InceptionV3 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,81, F1-Skoru 0,86 olarak tespit edilmiştir.

InceptionV3 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [11]: train_result = model_InceptionV3.evaluate(train_generator)
         val_result = model_InceptionV3.evaluate(test_generator)

         model_InceptionV3_result = pd.DataFrame(zip(train_result, val_result),
                                    columns=['Train', 'Validation'],
                                    index=['Loss', "Accuracy"])

         model_InceptionV3_result
```

```
16/16 [==============================] - 32s 2s/step - loss: 0.2467 - acc: 0.8953
4/4 [==============================] - 7s 2s/step - loss: 0.3563 - acc: 0.8505
```

Out[11]:

	Train	Validation
Loss	0.246722	0.356283
Accuracy	0.895314	0.850467

Şekil 148. InceptionV3 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretmiştir.

```
In [13]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
y_pred = model_InceptionV3.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 3s 992ms/step
Confusion matrix:
 [[19 10]
 [ 4 31]]
Accuracy Score:  0.78125
Classification report:
              precision    recall  f1-score   support

           0       0.66      0.83      0.73        23
           1       0.89      0.76      0.82        41

    accuracy                           0.78        64
   macro avg       0.77      0.79      0.77        64
weighted avg       0.80      0.78      0.79        64
```

Şekil 149. InceptionV3 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,78, F1-Skoru 0,82 olarak tespit edilmiştir.

InceptionV3 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 6. InceptionV3 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
InceptionV3 (tip-1)	0.205166	0.926221	0.383610	0.841121	0.88	0.859375
InceptionV3 (tip-2)	0.338046	0.837488	0.395603	0.794393	0.86	0.8125
InceptionV3 (tip-3)	0.246722	0.895314	0.356283	0.850467	0.82	0.78125

Tablo 6'da InceptionV3 tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-1 modelin daha başarılı skorlar ürettiği görülmektedir. Bu sayede, normal veriler ile eğitim gören modelin daha başarılı sonuç verdiği yorumlanabilir.

3.1.7. ResNet50 Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, ResNet50 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

ResNet50 Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_ResNet50.evaluate(train_generator)
        val_result = model_ResNet50.evaluate(test_generator)

        model_ResNet50_result = pd.DataFrame(zip(train_result, val_result),
                                             columns=['Train', 'Validation'],
                                             index=['Loss', "Accuracy"])

        model_ResNet50_result

        16/16 [==============================] - 64s 4s/step - loss: 0.4627 - acc: 0.80
        56
        4/4 [==============================] - 14s 3s/step - loss: 0.5079 - acc: 0.7430
```

Out[7]:

	Train	Validation
Loss	0.462728	0.507936
Accuracy	0.805583	0.742991

Şekil 150. ResNet50 Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre iyi sayılamayacak derecede skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
        y_pred = model_ResNet50.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))

        2/2 [==============================] - 5s 2s/step
        Confusion matrix:
         [[12 10]
         [11 31]]
        Accuracy Score:  0.671875
        Classification report:
                      precision    recall  f1-score   support

                   0       0.55      0.52      0.53        23
                   1       0.74      0.76      0.75        41

            accuracy                           0.67        64
           macro avg       0.64      0.64      0.64        64
        weighted avg       0.67      0.67      0.67        64
```

Şekil 151. ResNet50 Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,67, F1-Skoru 0,75 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermemiştir.

ResNet50 Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_ResNet50.evaluate(train_generator)
        val_result = model_ResNet50.evaluate(test_generator)

        model_ResNet50_result = pd.DataFrame(zip(train_result, val_result),
                                             columns=['Train', 'Validation'],
                                             index=['Loss', "Accuracy"])

        model_ResNet50_result

        16/16 [==============================] - 65s 4s/step - loss: 0.5097 - acc: 0.79
        16
        4/4 [==============================] - 14s 3s/step - loss: 0.5542 - acc: 0.7243
```

Out[7]:

	Train	Validation
Loss	0.509743	0.554223
Accuracy	0.791625	0.724299

Şekil 152. ResNet50 Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine iyi sayılamayacak derecede skorlar üretmiştir.

```
In [9]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
        y_pred = model_ResNet50.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))

        2/2 [==============================] - 5s 2s/step
        Confusion matrix:
         [[12 12]
         [ 6 34]]
        Accuracy Score:  0.71875
        Classification report:
                      precision    recall  f1-score   support

                   0       0.50      0.67      0.57        18
                   1       0.85      0.74      0.79        46

            accuracy                           0.72        64
           macro avg       0.68      0.70      0.68        64
        weighted avg       0.75      0.72      0.73        64
```

Şekil 153. ResNet50 Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,71, F1-Skoru 0,79 olarak tespit edilmiştir.

ResNet50 Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [8]: train_result = model_ResNet50.evaluate(train_generator)
        val_result = model_ResNet50.evaluate(test_generator)

        model_ResNet50_result = pd.DataFrame(zip(train_result, val_result),
                                             columns=['Train', 'Validation'],
                                             index=['Loss', "Accuracy"])

        model_ResNet50_result

16/16 [==============================] - 66s 4s/step - loss: 0.4643 - acc: 0.80
46
4/4 [==============================] - 14s 3s/step - loss: 0.5064 - acc: 0.7477
```

Out[8]:

	Train	Validation
Loss	0.464292	0.506417
Accuracy	0.804586	0.747664

Şekil 154. ResNet50 Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre başarılı skorlar üretememiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
         y_pred = model_ResNet50.predict(test_generator[0][0])
         y_pred = np.argmax(y_pred, axis=-1)

         y_test = test_generator[0][-1]

         #print(y_pred.shape)
         #print(y_test.shape)
         print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
         print("Accuracy Score: ", accuracy_score(y_test,y_pred))
         print("Classification report:\n", classification_report(y_pred,y_test))

2/2 [==============================] - 5s 2s/step
Confusion matrix:
 [[10  9]
 [ 9 36]]
Accuracy Score:  0.71875
Classification report:
              precision    recall  f1-score   support

           0       0.53      0.53      0.53        19
           1       0.80      0.80      0.80        45

    accuracy                           0.72        64
   macro avg       0.66      0.66      0.66        64
weighted avg       0.72      0.72      0.72        64
```

Şekil 155. ResNet50 Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,71, F1-Skoru 0,8 olarak tespit edilmiştir.

ResNet50 Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 7. ResNet50 temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
ResNet50 (tip-1)	0.462728	0.805583	0.507936	0.742991	0.75	0.671875
ResNet50 (tip-2)	0.509743	0.791625	0.554223	0.724299	0.79	0.71875
ResNet50 (tip-3)	0.464292	0.804586	0.506417	0.747664	0.80	0.71875

Tablo 7'de Xception tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa Tip-1, Tip-2 ve Tip-3 başarılı sayılamayacak derecede kötü skorlar üretmiştir.

3.1.8. Xception Temelli Modelin Eğitim-Test Sonuçları ve Metrikleri

Çalışmada, ResNet50 baz alarak üretilen Tip-1, Tip-2 ve Tip-3 modellerinin eğitim-test sonuçları ve metrikleri üretilmiştir.

Xception Temelli Tip-1 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [6]: train_result = model_Xception.evaluate(train_generator)
        val_result = model_Xception.evaluate(test_generator)

        model_Xception_result = pd.DataFrame(zip(train_result, val_result),
                                             columns=['Train', 'Validation'],
                                             index=['Loss', "Accuracy"])

        model_Xception_result

        16/16 [==============================] - 62s 4s/step - loss: 0.2993 - acc: 0.87
        04
        4/4 [==============================] - 13s 3s/step - loss: 0.3744 - acc: 0.7991
```

Out[6]:

	Train	Validation
Loss	0.299349	0.374359
Accuracy	0.870389	0.799065

Şekil 156. Xception Tip-1 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre iyi sayılamayacak derecede skorlar üretmiştir.

```
In [8]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
        y_pred = model_Xception.predict(test_generator[0][0])
        y_pred = np.argmax(y_pred, axis=-1)

        y_test = test_generator[0][-1]

        #print(y_pred.shape)
        #print(y_test.shape)
        print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
        print("Accuracy Score: ", accuracy_score(y_test,y_pred))
        print("Classification report:\n", classification_report(y_pred,y_test))

2/2 [==============================] - 5s 2s/step
Confusion matrix:
 [[20  4]
 [10 30]]
Accuracy Score:  0.78125
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.67      0.74        30
           1       0.75      0.88      0.81        34

    accuracy                           0.78        64
   macro avg       0.79      0.77      0.78        64
weighted avg       0.79      0.78      0.78        64
```

Şekil 157. Xception Tip-1 modeli metrikleri

Modelin doğruluk skoru 0,81, F1-Skoru 0,78 olarak tespit edilmiştir. Modelin doğruluk skoru ve F1-Skoru USB Bellek içeren görüntüleri sınıflandırma problemi için iyi sonuçlar vermemiştir.

Xception Temelli Tip-2 Modelin Eğitim-Test Sonuçları ve Metrikleri

```
In [7]: train_result = model_Xception.evaluate(train_generator)
        val_result = model_Xception.evaluate(test_generator)

        model_Xception_result = pd.DataFrame(zip(train_result, val_result),
                                             columns=['Train', 'Validation'],
                                             index=['Loss', "Accuracy"])

        model_Xception_result

16/16 [==============================] - 63s 4s/step - loss: 0.2474 - acc: 0.8973
4/4 [==============================] - 14s 3s/step - loss: 0.3297 - acc: 0.8411
```

Out[7]:

	Train	Validation
Loss	0.247448	0.329666
Accuracy	0.897308	0.841121

Şekil 158. Xception Tip-2 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre iyi skorlar üretmiştir.

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_Xception.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 5s 2s/step
Confusion matrix:
 [[14  7]
 [ 3 40]]
Accuracy Score:  0.84375
Classification report:
              precision    recall  f1-score   support

           0       0.67      0.82      0.74        17
           1       0.93      0.85      0.89        47

    accuracy                           0.84        64
   macro avg       0.80      0.84      0.81        64
weighted avg       0.86      0.84      0.85        64
```

Şekil 159. Xception Tip-2 modeli metrikleri

Modelin doğruluk skoru 0,84, F1-Skoru 0,89 olarak tespit edilmiştir. Model oldukça iyi sonuçlar üretmiştir.

Xception Temelli Tip-3 Modelin Eğitim-Test Sonuçları ve Metrikleri

In [7]:
```
train_result = model_Xception.evaluate(train_generator)
val_result = model_Xception.evaluate(test_generator)

model_Xception_result = pd.DataFrame(zip(train_result, val_result),
                                     columns=['Train', 'Validation'],
                                     index=['Loss', "Accuracy"])

model_Xception_result
```

```
16/16 [==============================] - 63s 4s/step - loss: 0.2474 - acc: 0.8973
4/4 [==============================] - 14s 3s/step - loss: 0.3297 - acc: 0.8411
```

Out[7]:

	Train	Validation
Loss	0.247448	0.329666
Accuracy	0.897308	0.841121

Şekil 160. Xception Tip-3 eğitim-test verilerine göre doğruluk ve hata skorları

Model eğitim-test verilerine göre oldukça başarılı skorlar üretmiştir.

```
In [10]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model_Xception.predict(test_generator[0][0])
y_pred = np.argmax(y_pred, axis=-1)

y_test = test_generator[0][-1]

#print(y_pred.shape)
#print(y_test.shape)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test,y_pred))
print("Classification report:\n", classification_report(y_pred,y_test))
```

```
2/2 [==============================] - 5s 2s/step
Confusion matrix:
 [[25  4]
 [ 3 32]]
Accuracy Score:  0.890625
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.89      0.88        28
           1       0.91      0.89      0.90        36

    accuracy                           0.89        64
   macro avg       0.89      0.89      0.89        64
weighted avg       0.89      0.89      0.89        64
```

Şekil 161. Xception Tip-3 modeli metrikleri

Modelin doğruluk skoru 0,89, F1-Skoru 0,9 olarak tespit edilmiştir. Modelin Tip-3 varyasyonu diğer varyasyonlarına göre daha başarılı olmuştur.

Xception Temelli Tip-1,Tip-2 ve Tip-3 modellerin Eğitim-Test Sonuçların ve Metriklerinin Karşılaştırılması

Tablo 8. Xception temelli Tip-1, Tip-2 ve Tip-3 modellerin eğitim-test sonuçları ve metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
Xception (tip-1)	0.299349	0.870389	0.374359	0.799065	0.81	0.78125
Xception (tip-2)	0.247448	0.897308	0.329666	0.841121	0.89	0.84375
Xception (tip-3)	0.258992	0.888335	0.333864	0.836449	0.90	0.890625

Tablo 8'de Xception tabanlı 3 tip modelin skorları görülmektedir. Skorlar baz alınarak yorumlanacak olursa veri arttırımı ve öğrenim aktarımı yaklaşımları sırasıyla uygulandığında modelin başarısı kademeli olarak artışa geçmiştir. Tip-3 modelinin ürettiği skorlar oldukça başarılıdır.

3.2. Elde Edilen Sonuçların İncelenmesi ve Yorumlanması

X-ray bagaj tarama sistemlerinden elde edilen görüntülerde USB bellek olup olmadığı sınıflandırması ile alakalı oluşturulan 8 farklı ESA bazlı modelin 3 tip varyasyonu ayrı ayrı incelendiğinde oldukça başarılı ve başarısız sonuçlar elde edilmiştir.

Tablo 9. Çalışılan modellerin tüm metrikleri

Çalışılan Model	loss	acc	val_loss	val_acc	f1-score	accuracy score
vgg16 (tip-1)	0,20343	0,907278	0,322865	0,850467	0,92	0,890625
vgg16 (tip-2)	0,23237	0,898305	0,328853	0,85514	0,92	0,890625
vgg16 (tip-3)	0,26058	0,886341	0,344587	0,82243	0,82	0,78125
vgg19 (tip-1)	0.154191	0.939182	0.261598	0.873832	0.92	0.890625
vgg19 (tip-2)	0.253939	0.887338	0.309147	0.831776	0.86	0.84375
vgg19 (tip-3)	0.194196	0.932203	0.262010	0.878505	0.94	0.921875
ResNet50V2 (tip-1)	0.236125	0.916251	0.357501	0.817757	0.90	0.875
ResNet50V2 (tip-2)	0.195320	0.905284	0.299025	0.883178	0.90625	0.93
ResNet50V2 (tip-3)	0.192718	0.912263	0.310100	0.873832	0.94	0.921875
MobileNetV2 (tip-1)	0.071029	0.979063	0.252285	0.873832	0.92	0.890625
MobileNetV2 (tip-2)	0.244411	0.879362	0.294663	0.836449	0.88	0.828125
MobileNetV2 (tip-3)	0.129258	0.969093	0.253112	0.887850	0.91	0.875
InceptionResNetV2 (tip-1)	0.219302	0.899302	0.318633	0.864486	0.92	0.890625
InceptionResNetV2 (tip-2)	0.318589	0.831505	0.351435	0.808411	0.85	0.8125
InceptionResNetV2 (tip-3)	0.254092	0.896311	0.326297	0.841121	0.86	0.8125
InceptionV3 (tip-1)	0.205166	0.926221	0.383610	0.841121	0.88	0.859375
InceptionV3 (tip-2)	0.338046	0.837488	0.395603	0.794393	0.86	0.8125
InceptionV3 (tip-3)	0.246722	0.895314	0.356283	0.850467	0.82	0.78125
ResNet50 (tip-1)	0.462728	0.805583	0.507936	0.742991	0.75	0.671875

Tablo 9’ un devamı

ResNet50 (tip-2)	0.509743	0.791625	0.554223	0.724299	0.79	0.71875
ResNet50 (tip-3)	0.464292	0.804586	0.506417	0.747664	0.80	0.71875
Xception (tip-1)	0.299349	0.870389	0.374359	0.799065	0.81	0.78125
Xception (tip-2)	0.247448	0.897308	0.329666	0.841121	0.89	0.84375
Xception (tip-3)	0.258992	0.888335	0.333864	0.836449	0.90	0.890625

Tablo 9’da bütün çalışılan modellerin varyasyonları ve üretilen metrikleri listelenmiştir. Tablo-9’dan edinilen bilgiye göre genel doğruluk skoru baz alınacak olursa en başarılı sonuç ResNet50V2 (tip-2) modelinde %93 genel doğruluk skoru ile alınmıştır. Tablo 9’daki skorlar yorumlanacak olursa en başarısız genel doğruluk skoru ResNet50 (tip-1) modelinden %67 genel doğruluk skoru ile alınmıştır.

Ayrıca yapılan çalışmada veri arttırımı tekniği ve buna bağlı olarak uygulanan öğrenim aktarımı tekniğinin uygulandığında başarılı sonuçlar verdiği tespit edilmiştir. Özellikle VGG19 ve Xception modellerinin varyasyonlarında öğrenim aktarımı tekniğinin işe yarar sonuçlar verdiği tespit edilip yüksek skorlar elde edilmiştir.

4. SONUÇLAR

Bu çalışmada, Evrişimsel Sinir Ağlarından oluşturulan modeller ile X-ray görüntüleme sistemlerinden elde edilen USB Bellek içeren görüntüler ile içermeyen görüntülerin sınıflandırılması çalışması yapılmıştır. Görüntü sınıflandırma alanında iyi sonuçlar vermiş başarılı mimariler mevcut problem üzerinde denenerek alınan sonuçlar kaydedilmiştir. X-ray görüntüleme sistemlerinin ürettiği RGB görüntülerin birçok alanda sınıflandırma uygulamaları yapıldığı halde veri güvenliği tehdidi oluşturan USB Bellek tespitiyle alakalı çalışmalara rastlanmamıştır. Bu sebeple yapılan tez çalışmasında aynı zamanda literatüre bir yenilik katmakta amaçlanmıştır.

Tez çalışmasında kullanılan veri kümeleri Rapiscan 620DV makinesi ile oluşturulmuştur. Görüntüler elde edildikten sonra derin öğrenme algoritmasının eğitim ve test aşamalarında faydalı olacağı düşünülmeyen görüntüler seçilip veri kümesinden atılmıştır. Ayrıca elde edilen görüntülere kesme işlemi uygulanmış görüntülerin boş olan alanları yazılan kod sayesinde kesilmiştir. Fakat bu veriler ile eğitilen modellerin çok başarısız olduğu tespit edildiği için çalışmada bu yaklaşıma yer verilmemiştir. Veri kümesi 1217 adet görüntüden oluşmakta olup bu görüntülerin 807 adeti USB Bellek içeren ve 410 adeti içermeyen olacak şekildedir. Ayrıca veri kümesi oluşturulurken X-ray görüntüleri farklı olan 8 ayrı USB Bellek ile çalışılmıştır.

Bilinen ESA mimarileri baz alınarak elde edilen modeller kendi içlerinde Tip-1, Tip-2 ve Tip-3 olacak şekilde varyasyonlara ayrılmıştır. Bütün varyasyonlar detaylı bir şekilde incelendiğinde çok başarılı, orta seviyede başarılı ve başarısız sonuçlar elde edilmiştir. Fakat 8 farklı mimari baz alınarak oluşturulan genel başarı skorlarına bakılırsa ResNet50V2, VGG19, VGG16, MobileNetV2 mimarilerinin oluşturduğu varyasyonlardan alınan skorlar USB Bellek içeren görüntülerin sınıflandırılması konusunda diğer mimarilerden daha başarılıdır. Ayrıca MobileNetV2'nin az parametreye sahip olması sebebi ile eğitim-test işlemlerinin çok kısa sürdüğü not edilmiştir.

Çalışmada öğrenim aktarımı tekniğinin başarılı sonuçlar verdiği tespit edilmiştir. Özellikle VGG19 ve Xception modellerinin oluşturduğu varyasyonlarda oldukça başarılı sonuçlar elde edilmiştir.

Derin öğrenme uygulamaları eğitilen ve test edilen veri miktarı arttırıldıkça daha başarılı sonuçlar üretebilmektedir. Çalışmada 1217 adet görüntü kullanılmıştır fakat

ilerleyen çalışmalarda daha fazla veri kullanılması hedeflenmektedir. Elde edilen sonuçlar uygulanan tekniklerin ve oluşturulan modellerin başarılı olduğunu söylemektedir fakat veri sayısı arttırılıp buna bağlı veri arttırımı ve öğrenme aktarımı tekniği uygulanırsa daha da başarılı sonuçlar elde edilebilir. Çünkü verinin çeşitlenmesi ve çoğaltılması işlemi, oluşturulan ve eğitilen modellerde daha başarılı sonuçlar vermektedir.

X-ray bagaj tarama sistemlerinin ürettiği görüntülerin sınıflandırılması özellikle güvenlik hassasiyeti duyan kurum ve kuruluşlar tarafından önemsenen ciddi bir konudur. Bu sebeple, derin öğrenme algoritmaları güvenlik ihlallerini asgari düzeye indirme konusunda iyi bir alternatif sunmaktadır. Derin öğrenme algoritmalarının hızlı ve yüksek doğruluk oranıyla görüntü taraması sayesinde kullanıcı gözünden kaçan tehlikeli nesneler kolayca tespit edilebilmektedir.

5. ÖNERİLER

Yapılan çalışmada, X-ray tarama cihazlarının ürettiği görüntülerden veri ihlaline sebebiyet verebilecek potansiyelde olan USB Belleklerin olduğu görüntülerin sınıflandırılması amaçlanmıştır. Bu amaç ile X-ray tarama cihazından çok sayıda veri üretilip eğitim-test işlemlerine hazır hale getirilmiştir. Oluşturulan farklı tipte ESA derin öğrenme mimarisine veri belli aralıklarla öğretilip sonuçlar değerlendirilmiştir. Benzer konularda çalışma yapmak isteyen araştırmacılara ve uygulayıcılara öneriler şu şekilde sıralanmıştır:

- Derin öğrenme mimarilerinde verinin fazla olması, oluşturulan modelin başarı performansını etkileyen önemli bir unsurdur. Bu sebeple hazırlanacak olan verilerin boyutunun arttırılması faydalı sonuçlar verebilir.
- Veri çeşitliliği derin öğrenme mimarilerinde model performansını arttıran başka bir önemli unsurdur. Bu amaçla daha çok USB Bellek tipi ile daha başarılı sonuçlar alınabilir.
- Bu yüksek lisans tez çalışmasında, bir görüntü sınıflandırma problemi üzerine çalışılmıştır. Görüntüler üzerindeki USB Bellek tespiti YOLOv3 veya Mask R-CNN derin öğrenme algoritmaları ile çalışılabilir.
- Daha fazla ön eğitimli model, ESA mimarisinin temeline koyularak veriler üzerinde farklı varyasyonlar denenebilir.

6. KAYNAKLAR

Abbas, A., Abdelsamea, M. M. & Gaber, M. M. (2021). Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. *Applied Intelligence*, 51, 854-864.

Ahmad, J., Muhammad, K. & Baik, S. W. (2017). Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. *PloS one*, 12(8), e0183838. doi: 10.1371/journal.pone.0183838

Ali, L., Alnajjar, F., Jassmi, H. A., Gocho, M., Khan, W. & Serhani, M. A. (2021). Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. *Sensors*, 21(5), 1688. doi:10.3390/s21051688

Akay, M., Du, Y., Sershen, C. L., Wu, M., Chen, T. Y., Assassi, S., Mohan, C. & Akay, Y. M. (2021). Deep learning classification of systemic sclerosis skin using the MobileNetV2 model. *IEEE Open Journal of Engineering in Medicine and Biology*, 2, 104-110.

Akcay, S. & Breckon, T. P. (2017, September). An evaluation of region based object detection strategies within x-ray baggage security imagery. In 2017 *IEEE International Conference on Image Processing (ICIP)* (pp. 1337-1341). IEEE.

Akgün, E. & Demir, M. (2018). Modeling course achievements of elementary education teacher candidates with artificial neural networks. *International Journal of Assessment Tools in Education*, 5(3), 491-509.

Bouvrie, J. (2006). Notes on convolutional neural networks.

Bre, F., Gimenez, J. M. & Fachinotti, V. D. (2018). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 1429-1441.

Das, K. & Behera, R. N. (2017). A survey on machine learning: concept, algorithms and applications. International *Journal of Innovative Research in Computer and Communication Engineering*, 5(2), 1301-1309.

Dave, V. S. & Dutta, K. (2014). Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, 42(2), 295-307.

Gad, A. F., Gad, A. F. & John, S. (2018). *Practical computer vision applications using deep learning with CNNs.* Berkeley: Apress.

Gaus, Y. F. A., Bhowmik, N. & Breckon, T. P. (2019, November). On the use of deep learning for the detection of firearms in x-ray baggage security imagery. In 2019 *IEEE International Symposium on Technologies for Homeland Security (HST)* (pp. 1-7). IEEE.

Griffin, T., Cao, Y., Liu, B. & Brunette, M. J. (2020, September). Object detection and segmentation in chest x-rays for tuberculosis screening. In 2020 *Second International Conference on Transdisciplinary AI (TransAI)* (pp. 34-42). IEEE.

Han, D., Liu, Q. & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, 95, 43-56.

Jain, D. K. (2019). An evaluation of deep learning based object detection strategies for threat object detection in baggage security imagery. *Pattern recognition letters*, 120, 112-119.

Ji, Q., Huang, J., He, W. & Sun, Y. (2019). Optimized deep convolutional neural networks for identification of macular diseases from optical coherence tomography images. *Algorithms*, 12(3), 51.

Kang, X., Song, B. & Sun, F. (2019). A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT. *Applied Sciences,* 9(1), 135.

Khattar, A. & Quadri, S. M. K. (2022). Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter. *Multimedia Tools and Applications*, 81(21), 30437-30464.

Lee, C. Y., Gallagher, P. W., & Tu, Z. (2016, May). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *In Artificial intelligence and statistics* (pp. 464-472). PMLR.

Le, T., Lee, M. Y., Park, J. R. & Baik, S. W. (2018). Oversampling techniques for bankruptcy prediction: Novel features from a transaction dataset. *Symmetry*, 10(4), 79.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S. & Zhang, G. (2015). Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80, 14-23.

Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F. & Zhang, Y. (2018). Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. *Remote Sensing*, 10(7), 1119.

Mamchur, D., Peksa, J., Le Clainche, S. & Vinuesa, R. (2022). Application and advances in radiographic and novel technologies used for non-intrusive object inspection. *Sensors*, 22(6), 2121.

Morchhale, S. (2016). *Deep convolutional neural networks for classification of fused hyperspectral and LiDAR data*. Wake Forest University.

Nair, R., Vishwakarma, S., Soni, M., Patel, T. & Joshi, S. (2022). Detection of COVID-19 cases through X-ray images using hybrid deep neural network. *World Journal of Engineering*, 19(1), 33-39.

Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O. & Acharya, U. R. (2020). Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in biology and medicine*, 121, 103792.

Prabhu, S., Naveen, D. K., Bangera, S. & Bhat, B. S. (2020, December). Production of X-Rays Using X-Ray Tube. *In Journal of Physics: Conference Series* (Vol. 1712, No. 1, p. 012036). doi: 10.1088/1742-6596/1712/1/012036

Ponnusamy, V., Marur, D. R., Dhanaskodi, D. & Palaniappan, T. (2021). Deep Learning-Based X-Ray Baggage Hazardous Object Detection-An FPGA Implementation. *Rev. d'Intelligence Artif.*, 35(5), 431-435.

Qazi, E. U. H., Zia, T. & Almorjan, A. (2022). Deep learning-based digital image forgery detection system. *Applied Sciences*, 12(6), 2851.

Saavedra, D., Banerjee, S. & Mery, D. (2021). Detection of threat objects in baggage inspection with X-ray images using deep learning. *Neural Computing and Applications*, 33, 7803-7819.

Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., ... & Rabczuk, T. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362, 112790.

Shi, B., Hou, R., Mazurowski, M. A., …. & Lo, J. Y. (2018, February). Learning better deep features for the prediction of occult invasive disease in ductal carcinoma in situ through transfer learning. In Medical imaging 2018: *computer-aided diagnosis* (Vol. 10575, pp. 620-625). SPIE. doi: 10.1117/12.2293594

Shorten, C. & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data,* 6(1), 1-48.

Teso Fernández de Betoño, A., Zulueta Guerrero, E., Cabezas Olivenza, M., Fernández Gámiz, U. & Botana Martínez de Ibarreta, C. (2023). Modification of Learning Ratio and Drop-Out for Stochastic Gradient Descendant Algorithm.

Toğaçar, M., Ergen, B. & Cömert, Z. (2020). COVID-19 detection using deep learning models to exploit Social Mimic Optimization and structured chest X-ray images using fuzzy color and stacking approaches. *Computers in biology and medicine*, 121, 103805.

Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global.

Venkatesan, R., & Li, B. (2017). *Convolutional neural networks in visual computing: a concise guide*. CRC Press.

Wang, Y., Lucas, M., Furst, J., Fawzi, A. A., & Raicu, D. (2020). Explainable deep learning for biomarker classification of oct images. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)* (pp. 204-210). IEEE.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data,* 3(1), 1-40.

Westphal, E. & Seitz, H. (2021). A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks. *Additive Manufacturing,* 41, 101965.

Wong, S. C., Gatt, A., Stamatescu, V. & McDonnell, M. D. (2016, November). Understanding data augmentation for classification: when to warp?. In *2016 international conference on digital image computing: techniques and applications* (DICTA) (pp. 1-6). IEEE.

Yingge, H., Ali, I., & Lee, K. Y. (2020). Deep neural networks on chip-a survey. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)* (pp. 589-592). IEEE.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *Advances in neural information processing systems*, 27.

Zhang, J., Song, X., Feng, J., & Fei, J. (2021). X-Ray image recognition based on improved Mask R-CNN algorithm. *Mathematical Problems in Engineering*, 2021, 1-14.

Zhu, X. J. (2005). Semi-supervised learning literature survey.

ÖZGEÇMİŞ

Ali HACIHAMZAOĞLU, ilköğretim öğrenimini Prof. İhsan Koz İlköğretim Okulu'nda, lise öğrenimini Kanuni Anadolu Lisesi'nde gördü. 2011 yılında Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği bölümünü kazandı. 1 yılı İngilizce hazırlık olan 5 yıllık lisans eğitimini 2016 yılında tamamladı. 2017 yılında kurduğu elektrik proje şirketinde 1,5 sene çalıştı. 2019 yılında Karadeniz Teknik Üniversitesinde yüksek lisans eğitimine başladı. B2 seviye İngilizce bilmektedir. 2020 yılının Şubat ayından itibaren Başaran İleri Teknoloji AselsanNet Bayii'nde savunma sanayii sektöründe elektrik-elektronik mühendisi olarak çalışmaktadır.