
GAZİ ÜNİVERSİTESİ
Gazi
1926

EVRİŞİMSEL SİNİR AĞLARINI KULLANARAK SAHTE YÜZ GÖRÜNTÜLERİNİN TESPİT EDİLMESİ

Emre ŞAFAK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MART 2023

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Emre ŞAFAK

30/03/2023

EVRİŞİMSEL SİNİR AĞLARINI KULLANARAK SAHTE YÜZ GÖRÜNTÜLERİNİN TESPİT EDİLMESİ

(Yüksek Lisans Tezi)

Emre ŞAFAK

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Mart 2023

ÖZET

Sahte yüz içeren görüntü ve videolar en yaygın dijital manipülasyon türüdür. Bu içerikler yanlış bilgilerin yayılmasına neden olarak birçok istenmeyen duruma sebep olabilir. Özellikle sahte yüz görüntüsü üretiminde makine öğrenmesi algoritmalarının kullanılması sahte içeriklerin ayırt edilmesini oldukça zorlaştırmıştır. Yüz manipülasyonları tüm yüz sentezi, kimlik değiştirme, nitelik manipülasyonu ve ifade değiştirme olmak üzere 4 temel gruba ayrılır. Yapılan çalışmada tüm yüz sentezi ile çekişmeli üretici ağlar kullanılarak üretilen sahte yüz görüntülerinin tespit edilebilmesi için hafif evrişimsel sinir ağları kullanılmıştır. Eğitim işleminde kullanılan veri setinde FFHQ veri setindeki 70.000 gerçek görüntü ile FFHQ veri seti kullanılarak StyleGAN2 ile üretilen 70.000 sahte görüntü yer almaktadır. Veri setinin %80'i eğitim %20'si test işlemi için kullanılmıştır. İlk olarak eğitim işlemi için MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile evrişimsel sinir ağları ayrı ayrı eğitilmiştir. Eğitim işleminde modeller ImageNet üzerinde ön eğitilmiş olarak transfer öğrenme ile tekrar kullanılmıştır. İlk eğitimler sonucunda EfficientNetB0 algoritması %93,64 ile en yüksek doğruluk oranına ulaşmıştır. Doğruluk oranını artırabilmek için en yüksek EfficientNetB0 algoritmasına transfer öğrenme için ReLU aktivasyon fonksiyonuna sahip iki yoğun katman (256 nöron), iki bırakma katmanı, bir düzleştirme katmanı, ReLU aktivasyon fonksiyonuna sahip bir yoğun katman (128 nöron) ve sınıflandırma için kullanılan softmax aktivasyon fonksiyonuna sahip iki düğümlü yoğun katman eklenmiştir. Bu işlem sonucu EfficientNetB0 algoritmasıyla %95,48 doğruluk oranına ulaşılmıştır. Son olarak %95,48 doğruluk oranına ulaşan model ile MobileNet ve MobileNetV2 modelleri yığınlama topluluk öğrenme yöntemi ile birlikte eğitilerek %96,44 ile en yüksek doğruluk oranına ulaşılmıştır.

Bilim Kodu : 92432

Anahtar Kelimeler : Sahte yüz tespiti, yüz manipülasyon tespiti, hafif evrişimsel sinir ağı modeli, derin öğrenme, transfer öğrenme, topluluk öğrenme, yığınlama topluluk öğrenme

Sayfa Adedi : 58

Danışman : Prof. Dr. Necaattin BARIŞÇI

DETECTION OF FAKE FACE IMAGES USING CONVOLUTIONAL NEURAL NETWORKS

(M. Sc. Thesis)

Emre ŞAFAK

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

March 2023

ABSTRACT

Images and videos containing fake faces are the most common type of digital manipulation. These contents can cause many negativities by causing fake information to be spread. The use of machine learning algorithms in the production of fake face images has made it very difficult to distinguish fake content. Face manipulations are divided into 4 basic groups; entire face synthesis, face identity manipulation (deepfake), facial attribute manipulation, and facial expression manipulation. In the study, lightweight convolutional neural networks were used to detect fake face images produced by using entire face synthesis and generative adversarial networks. The dataset used in the training process includes 70,000 real images in the FFHQ dataset and 70,000 fake images produced with StyleGAN2 using the FFHQ dataset. 80% of the dataset was used for training and 20% for testing. Firstly MobileNet, MobileNetV2, EfficientNetB0 and NASNetMobile convolutional neural networks are separately trained for the training process. In the training the models were pre-trained on ImageNet and reused with transfer learning. As a result of the first trainings EfficientNetB0 algorithm reached the highest accuracy of 93.64%. EfficientNetB0 algorithm revised to increase the accuracy rate two dense layers (256 neurons) with ReLU activation, two dropout layers, one flatting layer, one dense layer (128 neurons) with ReLU activation function and softmax activation function used for classification dense layer with two nodes. As a result of this process accuracy rate of 95.48% was achieved with EfficientNetB0 algorithm. Finally the model that reached 95.48% accuracy, MobileNet and MobileNetV2 models were trained together with the stacking ensemble learning method, and the highest accuracy rate was achieved with 96.44%.

Science Code : 92432

Key Words : Fake face detection, face manipulation detection, convolutional neural networks, deep learning, transfer learning, ensemble learning, stacking ensemble learning

Page Number : 58

Supervisor : Prof. Dr. Necaattin BARIŞÇI

TEŞEKKÜR

Tezimin yürütülmesi sırasında önerileri ile beni yönlendiren, bilgilendiren ve yaptığı katkılardan dolayı danışmanım Prof. Dr. Necaattin BARIŞÇI'ya, eğitim öğretim hayatım boyunca bende emeği olan hocalarıma ve bu süreçte bana destek olan aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
API	Uygulama Programlama Arayüzü (Application Programming Interface)
CPU	Merkezi İşlem Birimi (Central Processing Unit)
ESA	Evrişimsel Sinir Ağları
GPU	Grafik İşlem Birimi (Graphics Processing Unit)
LBPH	Yerel İkili Model Histogramı (Local Binary Pattern Histogram)
ReLU	Düzeltilmiş Doğrusal Birim(Rectified Linear Unit)
TPU	Tensor İşlem Birimi (Tensor Processing Unit)

1. GİRİŞ

Sahte görüntü ve video içerikleri internetin yaygın olarak kullanılmaya başlamasından itibaren insanları aldatmak veya eğlendirmek amacıyla üretilmektedir. Başlangıçta fotoğraf ve video düzenleme programları kullanılarak yapılan manipülasyonlar için son yıllarda yapay zeka algoritmaları kullanılmaya başlamıştır. Yapay zeka kullanarak sahte görüntü ve video içerikleri üretmek için genellikle otomatik kodlayıcılar ve çekişmeli üretici ağlar gibi derin öğrenme yöntemleri kullanılmaktadır [1]. Yüz görüntüleri manipülasyonu en yaygın olarak üretilen sahte görüntü türüdür. Sahte yüz görüntülerinin çoğu kişileri itibarsızlaştırma veya dolandırıcılık amaçlıdır. Sahte yüz görüntüleri ile kişiler bulunmadıkları yerlerde gösterilebilir veya kişilere söylemedikleri sözlerle konuşma yaptırılabilir. Sahte yüz manipülasyonları kimlik değiştirme, nitelik manipülasyonu, ifade değiştirme ve tüm yüz sentezi olmak üzere dört çeşittir [2].

Kimlik değiştirme manipülasyonu bir videodaki veya görüntüdeki kişinin yüz görüntüsünün başka bir kişi ile değiştirilmesidir. Seçilen yüz kaynak video veya görüntü içeriğinde gözlemlenen bir yüzle değiştirilir. Kimlik değiştirme manipülasyonu için genellikle bilgisayar grafiği temelli yöntemler (face swap uygulamaları) ve derin öğrenme (deepfake) algoritmaları kullanılmaktadır. Derin sahte temelli yaklaşımda kaynak ve hedef yüzün eğitim görüntülerini yeniden oluşturmak için eğitilmiş iki otomatik kodlayıcıya dayanır. Görüntüleri kırpmak ve hizalamak için bir yüz dedektörü kullanılır. Sahte bir görüntü/video oluşturmak için kaynak yüzün eğitilmiş kodlayıcısı ve kod çözücüsü hedef yüze uygulanır. Otomatik kodlayıcı çıktısı daha sonra Poisson görüntü düzenleme yöntemi kullanılarak görüntünün geri kalanıyla uyumlu hale getirilir [3]. Nitelik manipülasyonu, yüzün fiziksel özelliklerinin değiştirilmesidir. Nitelik manipülasyonu ile yüzün saç rengi, cilt rengi, yaş, cinsiyet, göz vb. özellikleri değiştirilebilir. Bu manipülasyon yöntemi için genellikle StarGAN adı verilen çekişmeli üretici ağlar kullanılır. StarGAN [4] bir ayırıcı (D) ve bir üreticiden (G) oluşur. Ayırıcı, bir giriş görüntüsünün sahte mi gerçek mi olduğunu tahmin etmeye çalışır ve gerçek görüntüyü karşılık gelen etki alanına göre sınıflandırır. Üretici, hem görüntü hem de hedef alan etiketini girdi olarak alır ve sahte bir görüntü oluşturur. Üretici, ayırıcı tarafından sağlanan orijinal etki alanı etiketi ile verilen sahte görüntüden orijinal görüntüyü yeniden oluşturmaya çalışır. Son olarak üretici, gerçek görüntülerden ayırt edilemeyen ve ayrımcı tarafından hedef alan olarak sınıflandırılabilen görüntüler üretmeye çalışır. Yaygın kullanıma sahip FaceApp mobil uygulaması kullanıcının nitelik manipülasyonu yapabilmesini sağlayan örnek uygulamalardan biridir. Kullanıcılar bu teknolojiyi sanal bir ortamda çeşitli ürünleri (kozmetik, makyaj, saç modeli vb.) denemek için kullanabilirler [2]. İfade değiştirme manipülasyonu, yüzün niteliklerinin değiştirilmesidir. Yüz canlandırma olarak da bilinen ifade değiştirme manipülasyonu ile yüzdeki ifadeleri (mutlu, heyecanlı, şaşkın, kızgın vb.)

değiştirilebilir. Bu yöntemle görüntüler canlandırılarak istenilen şekilde kullanılabilir. Örneğin ifade değiştirme manipülasyonu ile bir kişi hiç bulunmadığı bir yerde konuşma yapabilir. Bu sebeple ifade değiştirme manipülasyonu ciddi sonuçlar doğurabilir [2]. İfade değiştirme manipülasyonu için Face2Face [5] ve Neural Nextures [6] yöntemleri kullanılmaktadır. Tüm yüz sentezi, daha önce bulunmayan yüz görüntülerinin çekişmeli üretici ağlar kullanılarak üretilmesidir. Tüm yüz sentezi manipülasyonu için genellikle StyleGAN [7] gibi güçlü çekişmeli üretici ağlar kullanılmaktadır. StyleGAN, NVIDIA araştırmacıları tarafından geliştirilmiş gerçekçi sahte yüz görüntüleri üretebilmeyi sağlayan çekişmeli üretici ağdır. Tüm yüz sentezi ile yüksek düzeyde gerçeklikle yüz görüntüleri üretilebilmektedir. Tüm yüz sentezi oyun, 3 boyutlu modelleme, medya vb. birçok sektörlerde fayda sağlayabilirken yanlış bilgi yaymak için sosyal ağlarda çok gerçekçi sahte profiller oluşturmak için de kullanılabilir [3].

Yüz manipülasyonlarında yapay zeka algoritmalarının kullanılmaya başlanmasıyla gerçek içerikler ile sahte içeriklerin ayırt edilebilmesi oldukça zorlaşmıştır. Sahte yüz görüntüsü sayısının ve kalitesinin artması nedeniyle tespit edilmesini sağlamak amacıyla çalışmalar yapılmaya başlamıştır. Yüksek performansla çalışan sahte yüz görüntüsü algılama yöntemlerinin çoğu derin öğrenme temellidir. Genellikle evrişimsel sinir ağı veya tekrarlayan sinir ağı teknikleri kullanılmaktadır. Sahte yüz tespitinin yüksek doğrulukta yapılabilmesinin yanında kaynakların etkili kullanılması gerekmektedir. Geleneksel yaklaşımlarda veri kaynaklarından toplanan veriler merkezi sunucular üzerinde işlenmektedir. Veri miktarının giderek artması ve nesnelerin interneti ile milyarlarca cihazın internete bağlanmasıyla mevcut işlem kapasiteleri ve bant genişlikleri yeterli olmayacaktır [8]. Bu nedenle geliştirilecek sahte yüz tespiti modelinin bir merkezi işlem sunucusu yerine düşük işlem gücüne sahip mobil cihazlar üzerinde maksimum performans ile çalışabilmesi gerekmektedir. Yapılan çalışmada çekişmeli üretici ağlar tarafından üretilen sahte yüz görüntülerinin mobil cihazlarda çalışabilen MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile hafif evrişimsel sinir ağları kullanılarak tespit edilmesi sağlanmıştır. Eğitim işlemi için 70.000 gerçek ve 70.000 sahte görüntü içeren veri seti kullanılmıştır. EfficientNetB0 algoritması yapılan eğitim işlemleri sonucunda en yüksek doğruluk oranına ulaşmıştır. Doğruluk oranını artırabilmek için EfficientNetB0, MobileNet ve MobileNetV2 yığınlama topluluk öğrenme yöntemi ile birlikte eğitildiğinde %96,44 ile en yüksek doğruluk oranına ulaşılmıştır. Bu tez çalışmasının 2. bölümünde literatürde yer alan ilgili çalışmalar, 3. bölümünde yapay sinir ağları, 4. bölümünde evrişimsel sinir ağları, 5. bölümünde topluluk öğrenme, 6. bölümünde aktarım öğrenme, 7. bölümünde araştırma bulguları ve 8. bölümünde sonuç ve öneriler anlatılmaktadır.

2. LİTERATÜR TARAMASI

Wang vd. [2] tarafından yapılan çalışmada sahte yüz görüntülerinin tespiti için evrişimsel sinir ağı modelinin katmanlarının nöron davranışlarının izlenmesine dayanan yeni bir yöntem önerilmektedir. Yapılan çalışmalar nöron kapsamı ve davranışlarının derin öğrenme sistemlerinin model üzerinden yapılan saldırılara (adversarial attack) karşı başarılı sonuçlar verdiğini göstermektedir. Bu çalışmada nöron davranışlarının izlenmesinin de benzer sonuçlar verdiği görülmüştür. Evrişimsel sinir ağı katmanlarında nöron aktivasyonunun uygulanması ile sahte kalıpların daha iyi tespit edilmesi sağlanmıştır. Kimlik değiştirme, nitelik manipülasyonu, ifade değiştirme ve tüm yüz sentezi manipülasyon türünün tespit edilmesinde deneysel sonuçlar önerilen yöntemin daha iyi sonuçlar verdiğini göstermiştir. Önerilen yöntem tüm yüz sentezi manipülasyon yönteminde FFHQ veri seti kullanılarak StyleGAN2 ile üretilen sahte yüz görüntüsü veri setinde %91,9 doğruluk oranına ulaşılmıştır.

St vd. [9] tarafından yapılan çalışmada sahte yüz görüntülerinin tespiti için FisherFace ve LBPH (Local Binary Pattern Histogram) algoritmaları birlikte kullanılmıştır. FisherFace yüz alanı boyutunu azaltmak için kullanılırken LBPH yüz görüntülerini sınıflandırmak için kullanılmıştır. FisherFace temel olarak FLDA (Fisher's Lineer Discriminant Analysis) yöntemine dayanmaktadır. FisherFace algoritmasının en önemli avantajı mevcut diğer algoritmalardan daha hızlı çalışması ve düşük hata oranına sahip olmasıdır. LBPH, yüz görüntülerinin özelliklerini çıkararak görüntüyü daha az hesaplama karmaşıklığı ile tanımaktadır. Sahte yüz görüntüleri veri seti olarak FFHQ, 100K-Faces, DFFD, CASIA-WebFace veri setleri ayrı ayrı kullanılmıştır ve karşılaştırılmıştır. Önerilen yöntem FFHQ veri seti üzerinde %94,92 doğruluk oranına ulaşmıştır.

Bang vd. [10] tarafından yapılan çalışmada sahte yüz görüntülerini tespit edebilmek için çift dikkat temelli ince ayar yapılmış sinir ağı mimarisi önerilmektedir. Önerilen yöntemde önceden eğitilmiş model performans ve sağlamlığı artırabilmek için ince ayar dönüştürücü, MobileNet blok V3 ve kanal dikkat modülü ile entegre edilmiştir. İnce ayar dönüştürücü kendine ait bir dikkat modülünden ve bir alt örnekleme katmanından oluşur. Bunun yanında ince ayar için daha az veri kullanılması sağlanırken sınıflandırıcının diğer evrişimsel sinir ağları ile kolayca entegre edilebilir hale getirilmesi sağlanmıştır. Kanal dikkat modülü sahte görüntülerin özellik alanını yakalamak için kullanılır. MobileNetV3 evrişimsel sinir ağı görüntüleri sınıflandırmak için kullanılmıştır. Önerilen yöntemi test etmek için FaceForensics++ veri seti ve çeşitli GAN tarafından oluşturulan veri setleri kullanılmıştır. 8 farklı çekişmeli üretici ağ üzerinde modelin etkinliği kontrol edilmiştir. Önerilen yöntem StyleGAN ile üretilen veri seti üzerinde %81,43 doğruluk oranınaulaşırken StyleGAN2 ile üretilen veri seti

üzerinde %83,64 doğruluk oranına ulaşmıştır.

Hu vd. [11] tarafından yapılan çalışmada sahte yüz görüntülerinin tespit edilebilmesi için sadece gözlerin kullanıldığı bir yöntem önerilmektedir. Yalnızca iki göz arasındaki tutarsız kornea speküler dikkate alınarak sahte yüz görüntüleri tespit edilmiştir. Normalde gerçek insan yüz görüntülerinin kornea vurguları birbiriyle ilişkiliyken sahte yüz görüntülerinde bu ilişki sağlanamamaktadır. Gerçek yüz görüntüsünde iki gözün önden duruşu kamera ile aynı yöndedir, gözler ışık kaynağından uzaktadır ve ortamdaki tüm ışık kaynakları her iki göze de görünür. Gerçek yüz görüntüleri için FFHQ veri seti kullanılırken sahte yüz görüntüleri için StyleGAN2 yöntemi ile oluşturulan veri seti kullanılmıştır. Önerilen yöntem yapılan test işlemi sonucu %94 doğruluk oranına ulaşmıştır.

Guo vd. [12] tarafından yapılan çalışmada sahte yüz görüntülerinin tespit edilebilmesi için düzensiz gözbebeği şekillerinin kullanıldığı bir yöntem önerilmektedir. Gerçek gözlerde göz bebekleri dairesel veya elips şeklindeyken çekişmeli üretici ağlar tarafından oluşturulan göz bebekleri ise düzensizdir. Önerilen yöntem ilk olarak giriş görüntüsündeki yüzü tanımlamak için bir yüz dedektörü kullanır. Göz bebeğinin sınırlarını tespit edebilmek için EfficientNet-B5 ile geliştirilmiş U-Net tabanlı bir model kullanılmaktadır. Ardından göz bebeğinin elips ile uyumu ve düzensizliğini hesaplamak için mesafe ölçüsü için yaygın olarak kullanılan BIoU kullanılmıştır. Bu hesaplama sonucuna göre görüntünün sahte veya gerçek olduğu tespit edilebilmektedir. Yapılan çalışmada gerçek yüz görüntüleri için FFHQ veri seti, sahte yüz görüntüleri için StyleGAN2 yöntemi ile oluşturulan görüntülerden oluşan 1600 görüntüden oluşan veri seti kullanılmıştır. Önerilen yöntem yapılan test işlemi sonucu %91 doğruluk oranına ulaşmıştır.

Sahte yüz tespiti ile ilgili yapılan çalışmaların özeti Çizelge 2.1'de sunulmuştur.

Çizelge 2.1. Sahte yüz tespiti ile ilgili yapılan çalışmalar

Çalışma	Teknik	Algoritma	Özellik	Veri Seti	Değerlendirme Metriği	Başarı Oranı
Wang vd. [2]	Derin öğrenme	Nöron izlemesine dayalı yeni model	Sahte yüz tespiti	FFHQ ve StyleGAN 2	Doğruluk	%91,9
St vd. [9]	Derin öğrenme	FisherFace	Sahte yüz tespiti	FFHQ	Doğruluk	%94,92
Bang vd. [10]	Derin öğrenme	MobileNetV 3	Sahte yüz tespiti	FFHQ ve StyleGAN 2	Doğruluk	%83,64
Hu vd. [11]	Görüntü işleme	Dlib kütüphanesi	Sahte yüz tespiti	FFHQ ve StyleGAN 2	Doğruluk	%94
Guo vd. [12]	Derin Öğrenme	EfficientNet-B5	Sahte yüz tespiti	FFHQ ve StyleGAN 2	Doğruluk	%91

3. YAPAY SİNİR AĞLARI

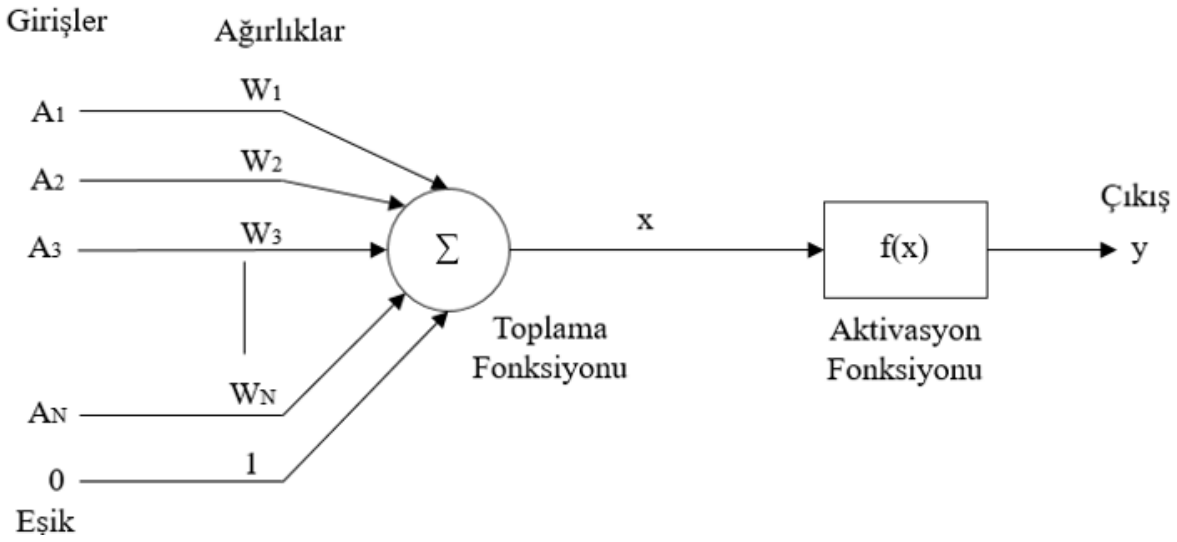
Yapay sinir ağları, sinir sistemimizdeki nöronlarla aynı temel prensibi paylaşır. Günümüzde en popüler makine öğrenme algoritmalarıdır. Yapay sinir ağları derin öğrenme olarak da bilinmektedir. Bilgileri giriş katmanından çıkış katmanına nöronlar aracılığıyla iletilmesini sağlar. Yapay sinir ağları öğrenebilir, tahmin veya sınıflandırma işlemi yapabilir. Görüntü tanıma, ses tanıma, çeviri ve tıbbi tanı gibi çeşitli uygulamalarda kullanılmaktadır. Yapay sinir ağlarının en büyük avantajı örnek veri modellerinden öğrenebilmesidir. Yapay sinir ağlarının gelişmiş tahmin yetenekleri sayesinde mevcut veri analiz teknikleri geliştirilebilir. Yapay sinir ağları 3 temel katmandan oluşmaktadır;

Giriş Katmanları (Input Layers): Giriş katmanı çeşitli metinler, sayılar, ses dosyaları, görüntü vb. bilgileri alan ilk yapay sinir ağı katmanıdır.

Gizli Katmanlar (Hidden Layers): Gizli katmanlar yapay sinir ağı modelinin ortasında yer almaktadır. Gizli katmanlar, giriş verileri üzerinde çeşitli matematiksel hesaplamalar gerçekleştirir ve bir eşleşmenin parçası olan kalıpların tanınmasını sağlar.

Çıktı Katmanı (Output Layer): Gizli katmanda gerçekleştiren hesaplamalar tamamlandıktan sonra çıktı katmanında işlem tamamlanarak eşleştirme yapılır.

Yapay sinir ağında yer alan tüm düğümlerin ağırlıkları vardır. Yapay sinir ağlarında çıktılar bazı parametrelere bağlıdır. Bu parametreler; ağırlıklar, önyargılar (biases), öğrenme oranı ve veri parti boyutudur. Hesaplanan ağırlıkların toplamı transfer fonksiyonu ile sağlanır. Düğüm ağırlıklarına bağlı olarak aktivasyon fonksiyonu uygun sonucu tetikler. Yapay sinir ağlarında kullanılan popüler aktivasyon fonksiyonlarından bazıları Sigmoid, RELU, Softmax, tanh fonksiyonlarıdır. En son aktivasyon fonksiyonunun sonucuna göre nihai çıktı elde edilir. Son olarak hata fonksiyonları kullanılarak tahmin edilen çıktı ve sonuçta ortaya çıkan çıktı arasındaki tutarsızlıklar hesaplanır ve sinir ağının ağırlıkları güncellenir. Bu ayarlama işlemine geri yayılım (backpropagation) adı verilir. Şekil 3.1'de yapay sinir ağı mimarisi gösterilmektedir [13].



Şekil 3.1. Yapay sinir ağı mimarisi [14]

Nöron Çıkışı = y = f(w1.A1 + w2.A2 + … + wN.AN +b) (1)

Girdiler (A1,A2,...,AN): Yapay sinir ağlarına dışarıdan veya diğer herhangi bir hücreden gelen verilerdir.

Ağırlıklar (w1,w2,...,wN): Hücreler arasındaki bağlantıların sayısal değerini ifade etmektedir. Bir hücreye gelen bilginin değerini ve hücre üzerindeki etkisini gösterir.

Toplama Fonksiyonu: Hücreye gelen girdileri ağırlıklarla çarpıp toplayarak o hücrenin net girdisinin hesaplanmasını sağlar.

Aktivasyon Fonksiyonu (f(x)): Aktivasyon fonksiyonunun amacı, bir nöronun çıkışına doğrusal olmama özelliği kazandırmaktır. Yapılan çalışmalarda genellikle sigmoid, tanh ve ReLU aktivasyon fonksiyonları kullanılmaktadır.

Sigmoid aktivasyon fonksiyonu aldığı girdiyi 0 ile 1 arasında bir değere dönüştürür. Sigmoid genellikle ikili sınıflandırma problemleri için kullanılmaktadır.

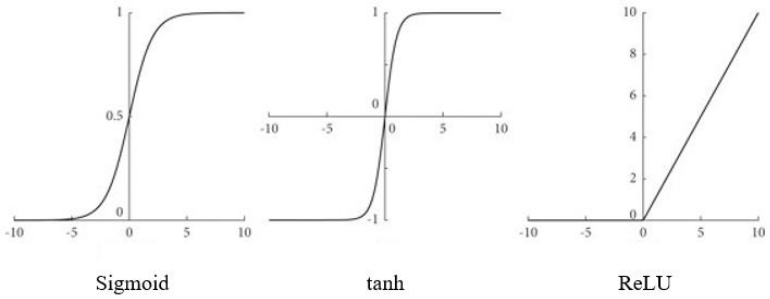
$$\sigma(x) = 1 / (1 + \exp(-x)) \quad (2)$$

tanh aktivasyon fonksiyonu aldığı girdiyi [-1, 1] aralığında bir değere dönüştürür.

$$\sigma(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x}) \quad (3)$$

ReLU aktivasyon fonksiyonu, negatif değerleri sıfıra eşitleyerek ve pozitif değerleri koruyarak ağa doğrusal olmama özelliği getirir. Bu sayede daha hızlı ve daha etkili eğitim yapılabilmesini sağlar. ReLU yerine kullanılabilecek tanh ve sigmoid fonksiyonlarından performans açısından daha etkilidir. Şekil 3.2'de aktivasyon fonksiyonları mimarisi gösterilmektedir [14].

f (x) = maks (0, x) (4)



Şekil 3.2. Aktivasyon fonksiyonları [13]

Çıktılar: Aktivasyon fonksiyonları tarafından üretilen değerlerdir. Çıktılar diğer kullanıcılara ya da başka bir hücreye gönderilebilir.

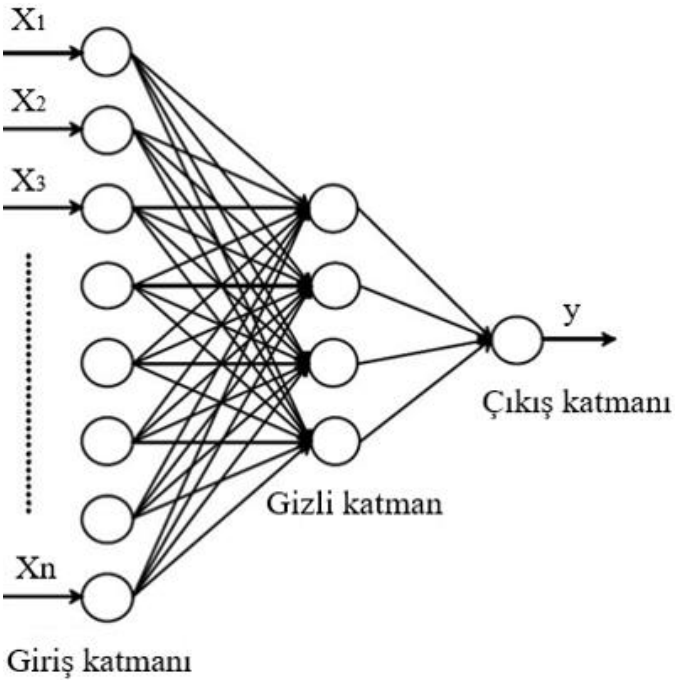
3.1. Yapay Sinir Ağı Türleri

Yapay sinir ağında eğitim işlemi tamamlandığında test verileri ile giriş çıkış eşleşmeleri kontrol edilir. Sinir ağı çıkışı tahmin eder ve farklı hata fonksiyonları kullanılarak çıktının ne kadar doğru olduğu değerlendirilir. Sonuca dayanarak ağı optimize etmek için sinir ağının ağırlıkları güncellenir.
Yapay sinir ağlarının iki türü vardır

- İleri beslemeli (feedforward) sinir ağı
- Geri beslemeli (feedback) sinir ağı

3.1.1. İleri beslemeli (feedforward) yapay sinir ağları

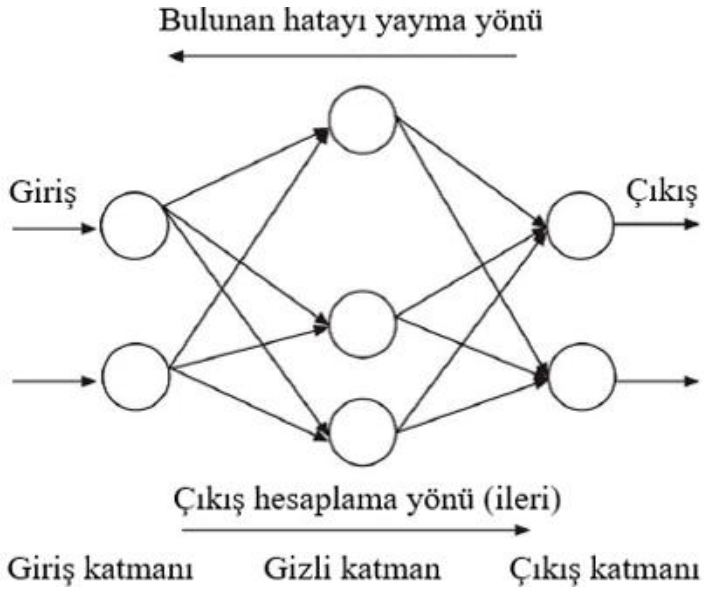
İleri beslemeli yapay sinir ağlarında bilgi akışı tek yönde gerçekleşir. Bilgi akışı giriş katmanından gizli katmana ve son olarak çıktıya kadardır. Bu sinir ağında geri bildirim döngüsü yoktur. Bu tür sinir ağları genellikle denetimli öğrenmede, sınıflandırmada ve görüntü tanıma durumlarında kullanılır. Şekil 3.3'de ileri beslemeli yapay sinir ağı mimarisi gösterilmektedir [15].



Şekil 3.3. İleri beslemeli yapay sinir ağı mimarisi [15]

3.1.2. Geri beslemeli (feedback) yapay sinir ağları

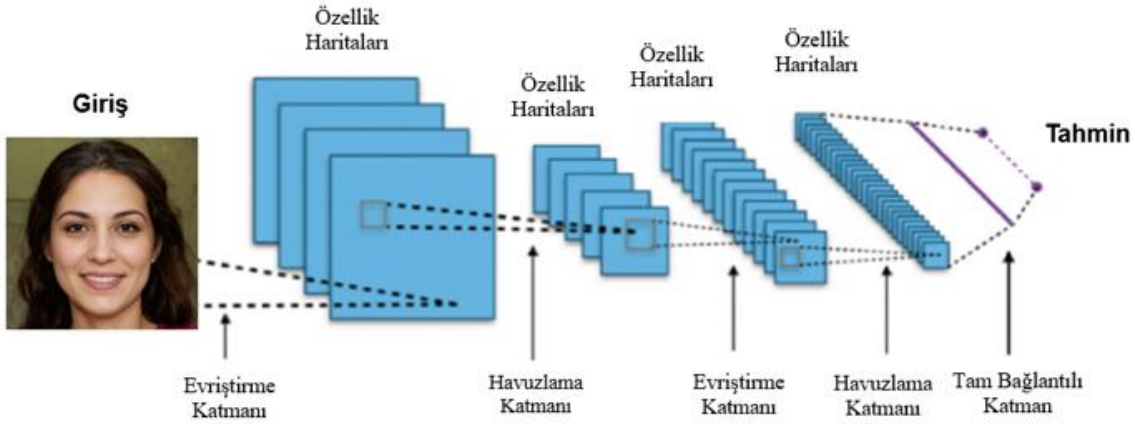
Geri beslemeli yapay sinir ağlarında geri besleme döngüleri kullanılmaktadır. Geri beslemeli yapay sinir ağında minimum bir hücrenin çıkışı kendisine veya diğer hücrelere girdi olarak verilir. Geri beslemeli yapay sinir ağları temel olarak hafızanın saklanması için kullanılmaktadır. Veriler sıralı veya zamana bağlı olduğu durumlarda kullanılır. Evrişimsel sinir ağları geri beslemeli yapay sinir ağıdır. Şekil 3.4'de geri beslemeli yapay sinir ağı mimarisi gösterilmektedir [16].



Şekil 3.4. Geri beslemeli yapay sinir ağı mimarisi [16]

4. EVRİŞİMSEL SİNİR AĞLARI

Evrişimsel sinir ağları ilk olarak 1980 yılında geliştirildi ve kullanıldı. Evrişimsel sinir ağları doğrudan verilerden öğrenen ve manuel özellik çıkarma ihtiyacını ortadan kaldıran derin öğrenmeye yönelik geliştirilmiş yapay sinir ağıdır. Evrişimsel sinir ağları bir girdi görüntüsünü alarak çeşitli bölgelere öğrenilebilir ağırlıklar atayarak öğrenme işlemini gerçekleştirir. Evrişimsel sinir ağları özellikle görüntü tanıma ve sınıflandırma görevlerinde oldukça başarılı sonuçlar vererek kendini kanıtlamıştır. Evrişimsel sinir ağları ses, zaman serileri, sinyal ve metin verileri gibi görüntü olmayan verileri tanıma ve sınıflandırma çalışmalarında da kullanılmaktadır. Belirli bir görev için geliştirilen evrişimsel sinir ağı başka bir görev için yeniden kullanılabilmektedir [16]. Evrişimsel sinir ağı mimarisinin çalışması Şekil 4.1'de sunulmuştur.



Şekil 4.1. Evrişimsel sinir ağının çalışması [16]

Şekil 4.1'de görüldüğü gibi alınan görüntü evrişimsel sinir ağı katmanlarından geçirilerek özellik haritası çıkarılır ve bu özellik haritalarına göre tahmin işlemi gerçekleştirilir.

4.1. Evrişimsel Sinir Ağı Mimarisi

Evrişimsel sinir ağları bir giriş ve çıkış katmanı arasında birçok gizli katmandan oluşur. En yaygın kullanılan gizli katmanlar; evrişim, aktivasyon fonksiyonu, havuzlama, bırakma ve tam bağlantılı katmandır [16].

4.1.1. Evrişim katmanı

Evrişim katmanı, evrişim filtrelerini kullanarak görüntü özelliklerinin tespit edilebilmesini sağlar. Filtreler 2x2, 3x3, 5x5 gibi farklı boyutlarda olabilir. Evrişim katmanı ilk olarak evrişim filtrelerini sinir ağından gelen girdilerle çarpar. Bu çarpma işlemi sırasında filtre görüntünün tamamını kaplamak için sağdan sola ve yukarıdan aşağıya uygulanarak görüntü üzerinden birçok kez geçilir.

Bu çarpımlar daha sonra toplanarak özellik haritaları oluşturulur [17]. Piksel değerleri yalnızca 0 ile 1 olan 5x5'lik görüntü matrisinin 3x3 matris ile evriştirilme işlemi Şekil 4.2'de görülmektedir.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Resim

1	0	1
0	1	0
1	0	1

Filtre

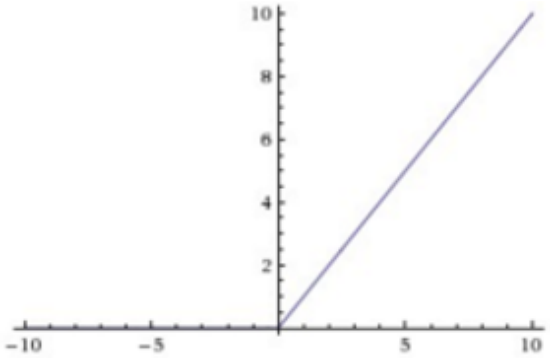
4	3	4
2	4	3
2	3	4

Özellik Haritası

Şekil 4.2. 5x5 giriş görüntüsüne 3x3'lük filtre uygulandığı evriştirme işlemi

4.1.2. Aktivasyon fonksiyonu katmanı

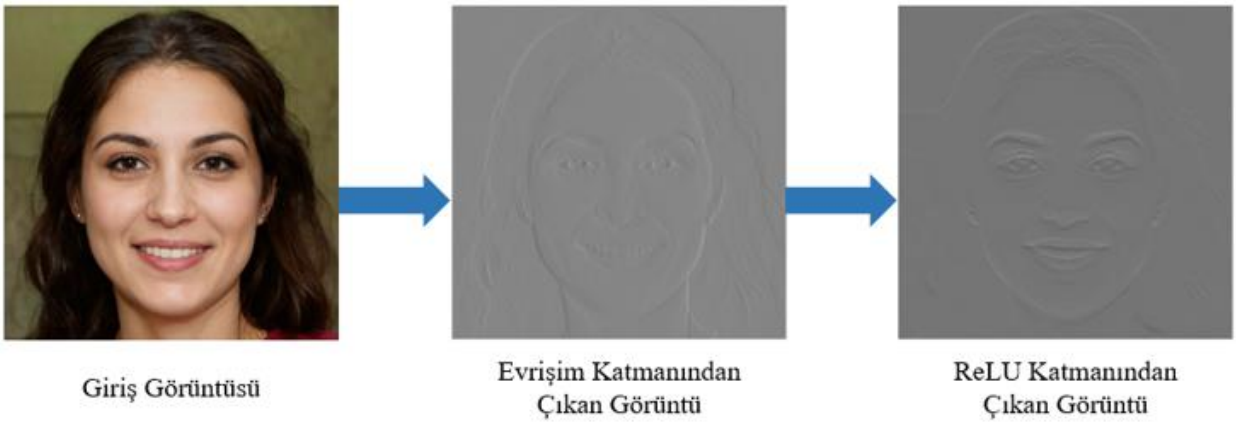
Aktivasyon Fonksiyonu katmanı, negatif değerleri sıfıra eşitleyerek ve pozitif değerleri koruyarak ağa doğrusal olmama özelliği getirir. Bu sayede daha hızlı ve daha etkili eğitim yapılabilmesini sağlar. ReLU yerine kullanılabilecek tanh ve sigmoid fonksiyonları da bulunmaktadır. Performans açısından ReLU bu iki fonksiyondan daha etkilidir [18]. ReLU işlemi Şekil 4.3'de görülmektedir.



Şekil 4.3. ReLU işlemi [16]

Çıkış = maksimum (sıfır, giriş) (5)

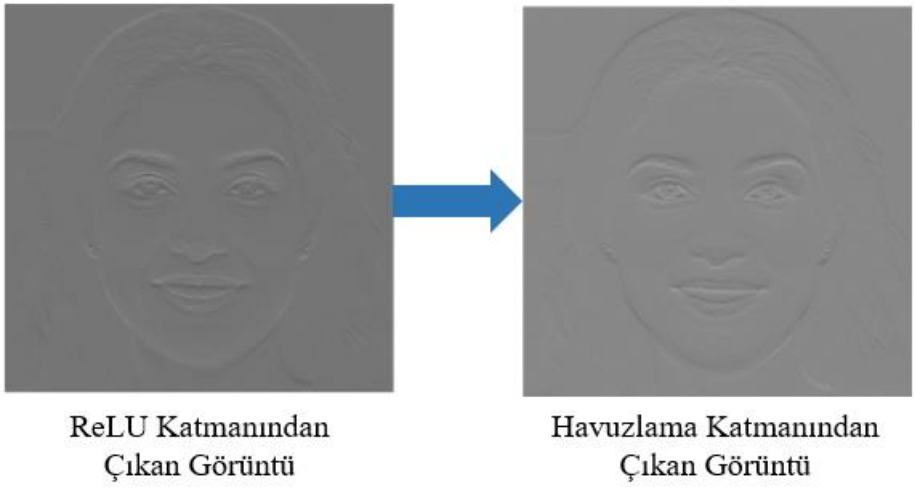
ReLU katmanının giriş görüntüsü üzerine yaptığı etki Şekil 4.4'de gösterilmiştir.



Şekil 4.4. Evriştirme katmanının ve ReLU katmanının giriş görüntüsüne yaptığı etki

4.1.3. Havuzlama katmanı

Havuzlama katmanı, yalnızca en önemli bilgileri tutarak girdideki parametre sayısını azaltır. Havuzlama katmanı ağdaki hesaplama ve parametre sayısını azaltarak fazla uyum oluşmasının engellenmesine, karmaşıklığı azaltmaya ve verimliliği artırmaya yardımcı olur. Girdi boyunca ağırlığı olmayan bir filtre gezdirilir. Filtreler, havuzlama türüne göre çıktı dizisini oluşturur. Maksimum ve ortalama havuzlama olmak üzere iki ana havuzlama türü vardır. Maksimum havuzlama yönteminde filtre giriş boyunca hareket ederken çıkış dizisine gönderilecek maksimum değere sahip pikseli seçer. Ortalama havuzlama yönteminde filtre giriş boyunca hareket ettikçe çıkış dizisine göndermek için alıcı alan içindeki ortalama değeri hesaplar [19]. Şekil 4.5'de Havuzlama ve ReLU katmanından çıkan giriş görüntüleri gösterilmektedir.



Şekil 4.5. Havuzlama katmanından çıkan görüntü

4.1.4. Bırakma katmanı

Bırakma katmanı, evrişimsel sinir ağının ezberleme yapmasını engellemek amacıyla rastgele seçilen belirli nöron setlerinin eğitim aşaması sırasında dikkate alınmamasını sağlar. Bırakma katmanı

kullanılmayan evrişimsel sinir ağlarında tüm ağırlıklar birlikte öğrenilmektedir. Bu durumda özelliklerin bir kısmı öğrenilirken diğer özellikler öğrenilmez ve ağ ezberlemeye başlar. Bırakma katmanı sayesinde ağdaki tüm ağırlıklar yerine bazı ağırlıkların öğrenilmesi sağlanır [20].

4.1.5. Tam bağlantılı katmanı

Tam bağlantılı katman önceki katmanlardan gelen öznitelikliere dayalı olarak sınıflandırma işlevini gerçekleştirir. Sınıflandırma işlemi için de softmax veya sigmoid gibi bir aktivasyon fonksiyonu kullanılmaktadır. Tam bağlantılı katman genellikle girdileri uygun bir şekilde sınıflandırabilmek için softmax aktivasyon işlevini kullanarak 0 ile 1 arasında olasılık üretir [21].

4.2. Evrişimsel Sinir Ağının Eğitilmesi

Evrişimsel sinir ağlarının eğitilmesi için genel olarak aşağıdaki işlemler sırasıyla gerçekleştirilir.

İşlem 1: Eğitim işlemi için ilk olarak veri seti hazırlama işlemi tamamlanır. Veri seti hazırlanırken veri toplama işlemi ile uygun veri kümesi oluşturulur. Ardından veri temizleme işlemi ile hatalı, tekrarlı, düşük kaliteli vb. veriler temizlenir.

İşlem 2: Evrişimsel sinir ağı mimarisi oluşturulur. Bu adımda mevcut bir model olduğu gibi kullanılabilir, mevcut model revize edilerek kullanılabilir veya tamamen yeni bir mimari geliştirilebilir. Kullanılacak model içerisinde yer alacak evrişim, aktivasyon fonksiyonu, havuzlama, bırakma ve tam bağlantılı katman sayıları ile sıralamaları belirlenir.

İşlem 3: Evrişimsel sinir ağı modeli oluşturulduktan sonra başlangıç değişkenleri tanımlanır. Bu değişkenler temel olarak filtre boyutu, filtre sayısı, öğrenme oranı, kayıp fonksiyonu ve adım sayısıdır.

İşlem 4: Evrişimsel sinir ağı görüntüleri alarak ilgili katmanlar üzerinden geçirir ve her bir sınıf için olasılıkları hesaplar. Görüntünün hangi sınıfa ait olduğu konusunda bir tahmin değeri üretir.

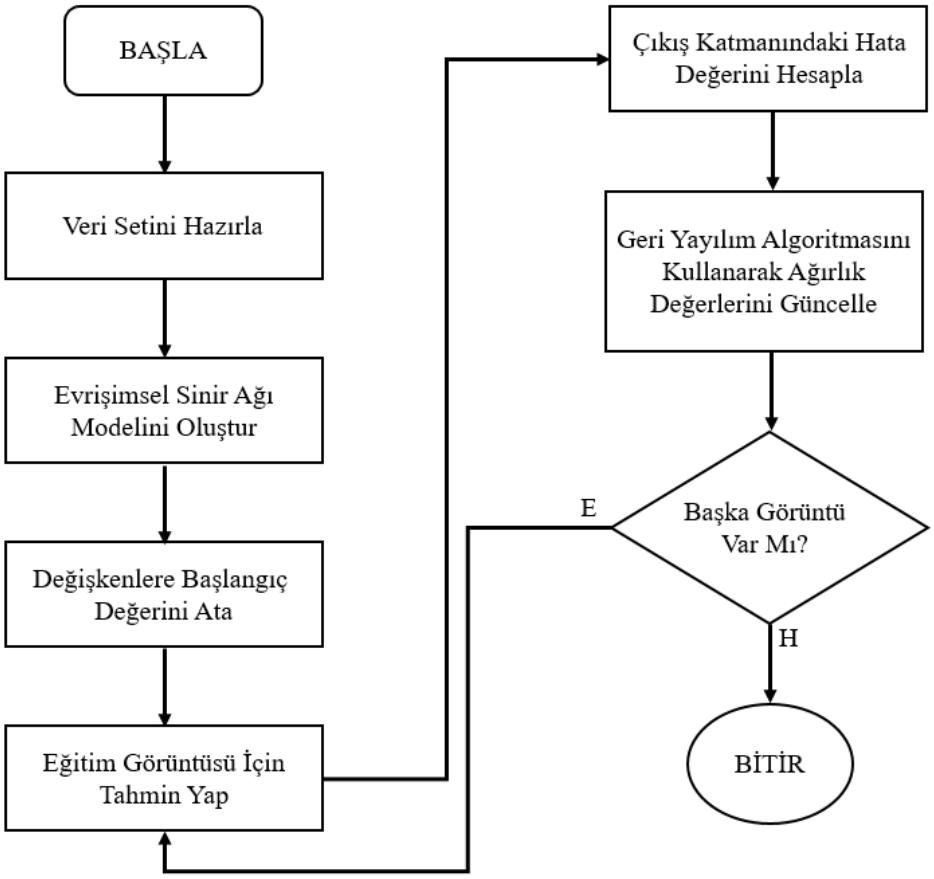
İşlem 5: Tahminin yapıldığı son katmanda hata değeri hesaplanır [21].

$$\text{Toplam Hata} = \sum 1/2 \,(\text{hedef olasılık} - \text{çıktı olasılığı})^{2} \quad (6)$$

İşlem 6: Evrişimsel sinir ağındaki ağırlıklara göre hatayı hesaplamak için geri yayılım algoritmaları kullanılır. Yapay sinir ağında eğitim işlemi tamamlandığında test verileri ile giriş çıkış eşleşmeleri kontrol edilir. Sinir ağı çıkışı tahmin eder ve farklı hata fonksiyonları kullanılarak çıktının ne kadar doğru olduğu değerlendirilir. Sonuca dayanarak ağı optimize etmek için geri yayılım algoritmaları kullanılarak sinir ağının ağırlıkları güncellenir. Sınıflandırma başarı oranının artırılması amaçlanmaktadır.

İşlem 7: Eğitim veri setindeki tüm görüntüler için 4 ve 6 arasındaki adımlar tekrarlanır.

Evrişimsel sinir ağının eğitilme adımlarının açıklandığı akış diyagramı Şekil 4.6'da görülmektedir.



Şekil 4.6. Evrişimsel sinir ağının eğitilmesi işleminin akış şeması

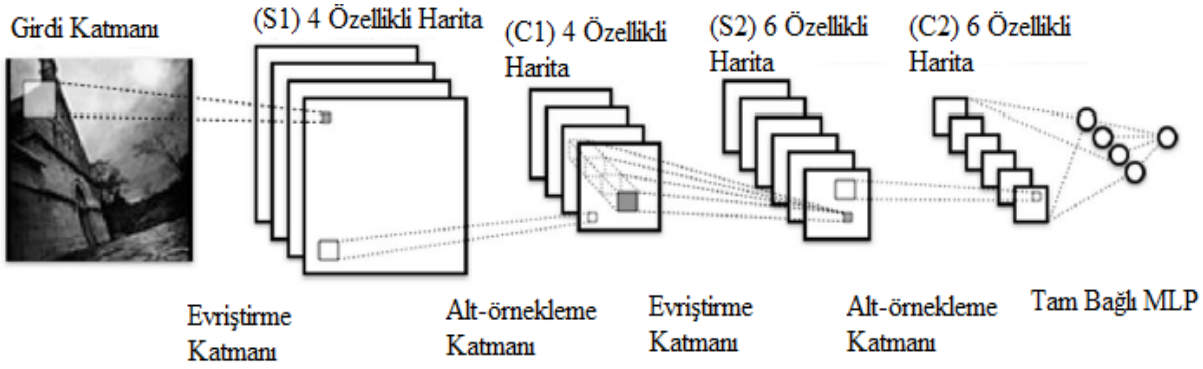
Şekil 4.6'da evrişimsel sinir ağının eğitim süreci gösterilmiştir. Eğitim setindeki her veri bu işlemlerden geçmektedir ve ağırlıklar yani yapılan tahminler geri yayılım algoritması sayesinde güncellenmektedir. Bu nedenle eğitim setindeki veri sayısı ne kadar fazla ve çeşitli ise modelin başarı oranı aynı oranda artacaktır.

4.3. Evrişimsel Sinir Ağı Modelleri

4.3.1. LeNet

LeCun ve ekibi tarafından 1998 yılında geliştirilen evrişimsel sinir ağıdır. İlk olarak el yazısı tanıma için kullanıldı ve diğer yöntemlerden daha iyi sonuçlar verdi. Ayrıca rakam tanıma, karakter tanıma gibi görevleri için de kullanılmıştır. LeNet, evrişimsel sinir ağının temelini oluşturan evrişim, havuzlama ve tam bağlantılı katmanlardan oluşmaktadır. Aktivasyon fonksiyonu olarak tanh aktivasyon fonksiyonu kullanılmıştır. LeNet mimarisi 3 evrişim katmanı, 2 havuzlama katmanı ve 2 tam bağlantılı katman olmak üzere toplam 7 katmandan oluşmaktadır.

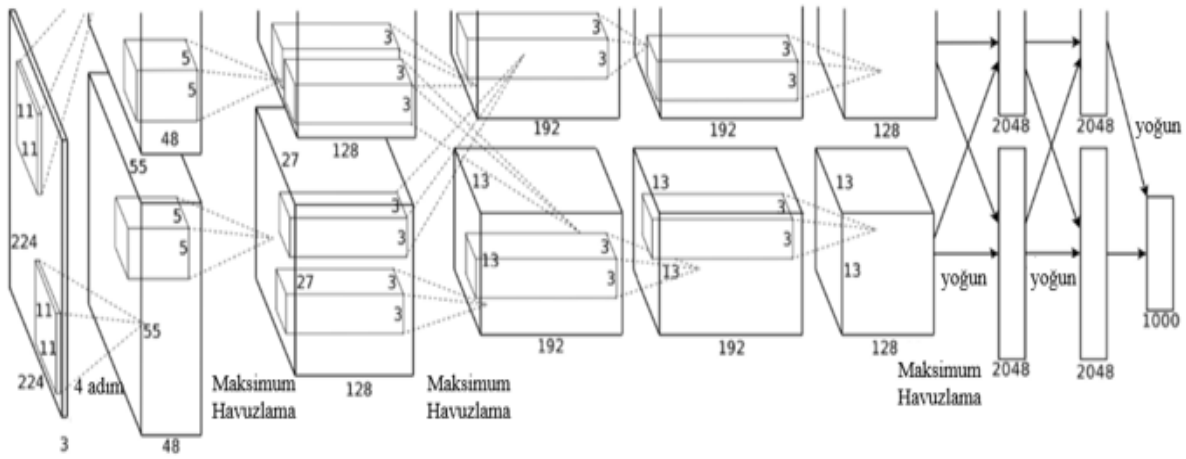
Günümüzde kullanılan evrişimsel sinir ağı modelleri LeNet ağını temel almaktadır [22]. LeNet mimarisi Şekil 4.7'de görülmektedir.



Şekil 4.7. LeNet mimarisi [22]

4.3.2. AlexNet

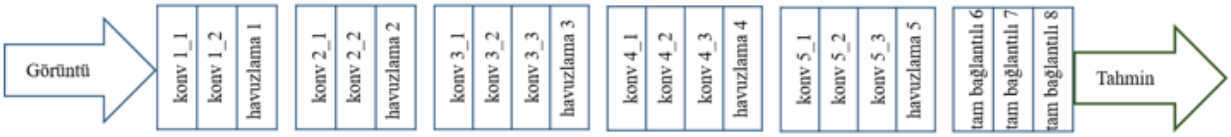
AlexNet, 2012 yılında ImageNet büyük ölçekli görsel tanıma yarışmasını kazanan evrişimsel sinir ağı mimarisidir. AlexNet, 2012 yarışmasını %15,3'lük ilk 5 hata oranıyla kazanmıştır. AlexNet, performansı artırmak için GPU kullanan ilk evrişimsel sinir ağıdır. AlexNet mimarisi 5 evriştirme katmanı, 3 maksimum havuzlama katmanı, 2 normalleştirme katmanı, 2 tamamen bağlı katman ve 1 softmax katmanından oluşur. Her evrişimli katman evrişimli filtrelerden ve doğrusal olmayan bir aktivasyon fonksiyonu ReLU'dan oluşur. AlexNet, ReLU doğrusal olmama özelliğini kullanarak evrişimsel sinir ağlarının tanh veya sigmoid gibi aktivasyon fonksiyonlarına göre çok daha hızlı eğitilebileceğini gösterdi. Bırakma tamamen bağlı 2 katmanda uygulanır. Bırakma sırasında 0,5 olasılıkla bir nöron düşürülür. Bir nöron düşürüldüğünde, ileriye doğru veya geriye doğru yayılmaya katkıda bulunmaz. Bırakma öğrenilen ağırlıkların daha güçlü olmasını sağlar ve ağın aşırı uyumunu engeller. AlexNet toplamda 60 milyon parametreye sahiptir [23]. AlexNet mimarisi Şekil 4.8'de görülmektedir.



Şekil 4.8. AlexNet mimarisi [23]

4.3.3. VGG

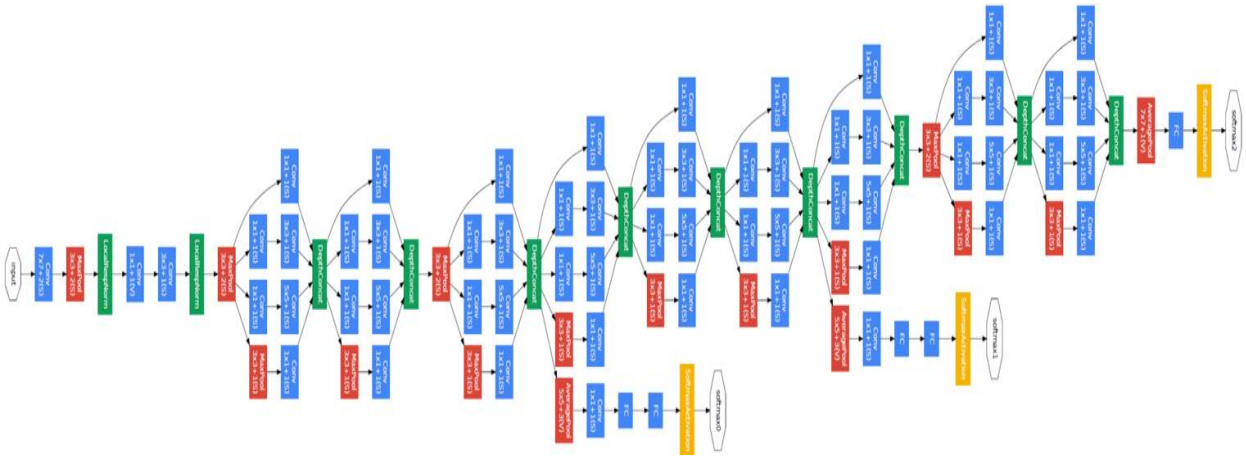
VGG, Simonyan ve Zisserman tarafından geliştirilmiş evrişimsel sinir ağıdır. VGG, evrişimsel sinir ağlarının daha derin olarak tasarlanmasına odaklanmaktadır. Ağda 3x3 boyutunda küçük filtreler kullanılır. Küçük boyutlu filtrelerin kullanılması parametre sayısını azaltarak karmaşıklığın giderilmesini sağlar. VGG16, 16 evrişim katmanından oluşurken VGG19 19 evrişim katmanından oluşmaktadır. VGG-19, VGG-16 evrişimsel sinir ağıyla aynı olup yalnızca 3 evrişim katmanı fazladır. VGG16 evrişimsel sinir ağı %92,7 ilk 5 test doğruluğu elde edilmiştir. VGG16 ağı toplamda yaklaşık 138 milyon parametreye sahiptir. VGG16 derinliği ve tamamen bağlı katman sayısı nedeniyle VGG16 modeli yaklaşık 533 MB boyutundadır. VGG evrişimsel sinir ağında tüm gizli katmanlar aktivasyon fonksiyonu olarak ReLU kullanmaktadır. VGG ağında 13 evriştirme katmanı ve 3 tamamen bağlı katman bulunmaktadır. VGG evrişimsel sinir ağı küçük boyutlu evrişim filtreleri kullanılması sayesinde katman sayısı ve performans artırılmıştır [24]. Yaygın olarak kullanılan ve en iyi performansa ulaşan VGG16 mimarisi Şekil 4.9'da görülmektedir.



Şekil 4.9. VGG16 mimarisi [24]

4.3.4. GoogleNet

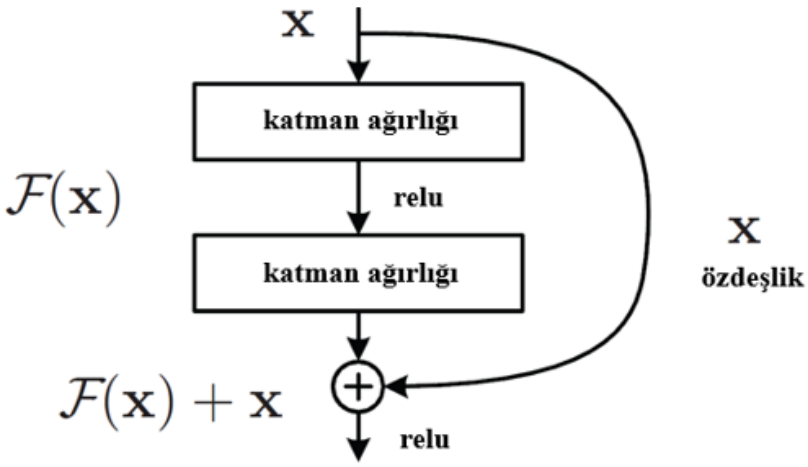
GoogleNet, Inception mimarisine dayalı evrişimsel sinir ağıdır. Inception mimarisi görüntülerde birden fazla filtre kullanılmasına dayalı modeldir. Inception mimarisinde 1×1, 3×3, 5×5 evriştirme ve 3×3 maksimum havuzlama aynı anda gerçekleştirilir. Farklı boyutlardaki evrişim filtrelerinin kullanılmasının amacı farklı boyutlardaki görüntülerin daha iyi işlenebilmesini sağlamaktır. GoogleNet daha önceki evrişimsel sinir ağlarından farklı olarak Inception mimarisinden gelen 1x1 evrişim filtrelerini kullanır. 1x1 evrişim filtreleri parametre sayısını azaltmak için kullanılır. GoogleNet mimarisi 22 katmandan oluşmaktadır. Parametre sayısı yaklaşık 4 milyondur. 2014 yılında ImageNet büyük ölçekli görsel tanıma yarışmasını kazanan evrişimsel sinir ağı mimarisidir. GoogleNet, 2014 yarışmasını %6,67'lik ilk 5 hata oranıyla kazanmıştır. GoogleNet evrişimsel sinir ağında aktivasyon fonksiyonu olarak ReLU kullanmaktadır. Ezberlemeyi engellemek için bırakma katmanı kullanılmaktadır. Bırakma katmanındaki bırakma oranı 0,7 olarak belirlenmiştir [25]. GoogleNet mimarisi Şekil 4.10'da görülmektedir.



Şekil 4.10. GoogleNet mimarisi [25]

4.3.5. ResNet

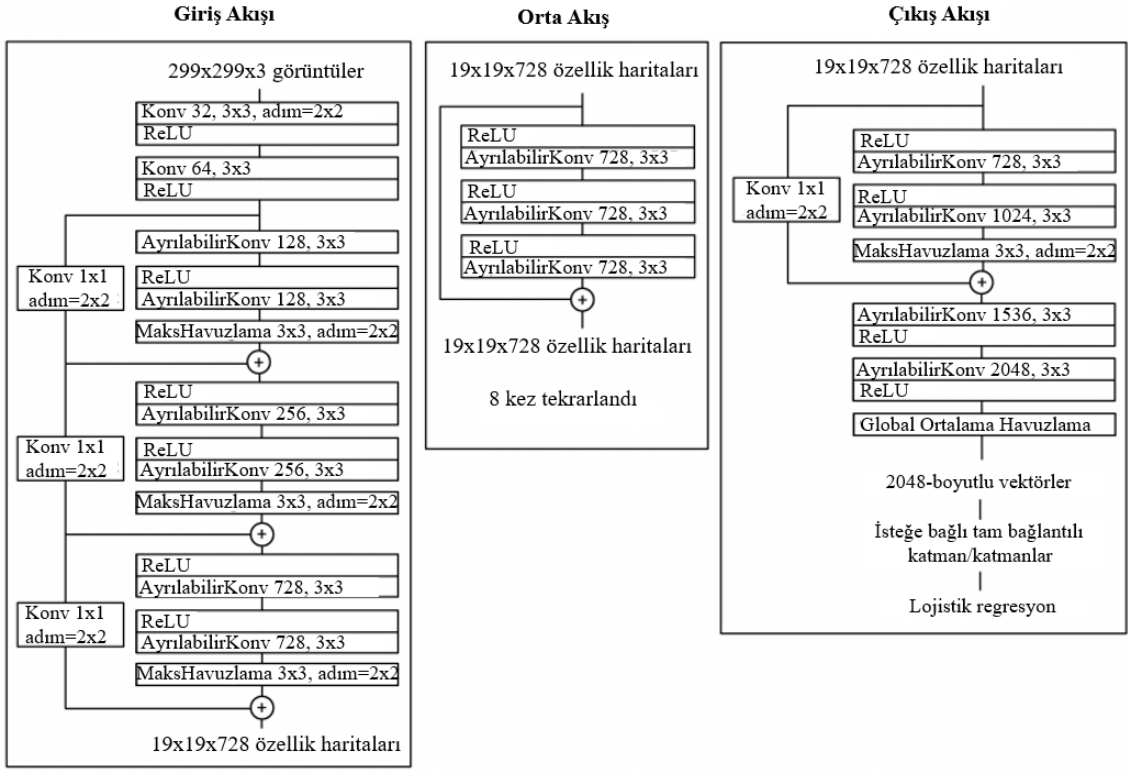
ResNet, 2015 yılında Kaiming He, Xiangyu Zhang, Shaoqing Ren ve Jian Sun tarafından geliştirilen evrişimsel sinir ağıdır. 2015 yılında ImageNet büyük ölçekli görsel tanıma yarışmasını kazanan evrişimsel sinir ağı mimarisidir. ResNet, 2015 yarışmasını %3,57'lik ilk 5 hata oranıyla kazanmıştır. Evrişimsel sinir ağlarında katman sayısı arttıkça eğitim hata oranı artabilmektedir. Evrişimsel sinir ağlarında geri yayılım sırasında kaybolan gradyan oluşur. ResNet, çok derin ağların eğitim sorunu olan kaybolan gradyan problemini çözmek için artık blokları kullanır. ResNet ağında eğitim işlemi sırasında modelin bazı katmanlarını atlayan doğrudan bir bağlantı vardır. Artık bağlantıların amacı daha başarılı gradyan akışına izin vermek ve önemli özelliklerin son katmanlara kadar taşınmasını sağlamaktır. ResNet evrişimsel sinir ağında aktivasyon fonksiyonu olarak ReLU kullanmaktadır. ResNet ağının 34, 50, 101 ve 152 katmanlı farklı versiyonları bulunmaktadır [26]. ResNet evrişimsel sinir ağının temelinde kullanılan artık blok mimarisi Şekil 4.11'de görülmektedir.



Şekil 4.11. Artık blok mimarisi [26]

4.3.6. Xception

Xception, keras yazılım kütüphanesinin geliştiricisi Google çalışanı François Chollet tarafından 2016 yılında geliştirilmiştir. Xception mimarisi çoğu klasik sınıflandırma mücadelesinde VGG-16, ResNet ve Inception V3'ten daha iyi performans gösterdi. Xception mimarisi Inception V3 ile aynı sayıda parametreye sahip olduğundan performans kazanımları artan kapasiteden değil, daha çok model parametrelerinin daha verimli kullanılmasından kaynaklanmaktadır Xception, 71 katmandan oluşan evrişimsel sinir ağıdır. Google tarafından önerilen önceki evrişimsel sinir ağı olan Inception mimarisinden farklı olarak Xception derinlemesine ayrılabilir evrişimleri kullanır. Derinlemesine ayrılabilir evrişimler hesaplama süresi açısından klasik evrişimlere göre çok daha verimlidir. Derinlemesine ayrılabilir evrişim yapısında filtrenin uzamsal ve derinlik boyutu ayrı olarak uygulanır. Xception ayrıca ResNet mimarisinde yer alan artık blokları da kullanmaktadır. Xcepton evrişimsel sinir ağında aktivasyon fonksiyonu olarak ReLU kullanmaktadır [27]. Xception mimarisi Şekil 4.12'de görülmektedir.



Şekil 4.12. Xception mimarisi [27]

4.4. Hafif Evrişimsel Sinir Ağı Modelleri

4.4.1. MobileNet

MobileNet, mobil ve gömülü görüntü uygulamaları için geliştirilmiş verimli çalışabilen ve hesaplama açısından çok yoğun olmayan evrişimsel sinir ağıdır. MobileNet 28 katman 4,2 milyon parametreden

oluşmaktadır. MobileNet, hafif evrişimsel sinir ağları oluşturmak için derinlemesine ayrılabilir evrişimler kullanır. MobileNet ağında derinlemesine ayrılabilir evrişimler kullanılarak parametre sayısı önemli ölçüde azaltılmıştır. Derinlemesine ayrılabilir evrişim derinlemesine ve noktasal evrişim olmak üzere iki katmandan oluşur. Derinlemesine evrişim her bir girişe tek bir filtre uygulamak için kullanılır. Derinlemesine evrişim yalnızca giriş kanalını filtrelemek için kullanıldığından yeni özellikler üretmek için bu filtreleri birleştiremez. Noktasal evrişim, derinlemesine evrişim çıktısının doğrusal bir kombinasyonunu hesaplayan 1x1 evrişimdir. MobileNet transfer öğrenmeyi çalıştırmak veya uygulamak için çok daha az hesaplama gücü gerektirir. MobileNet, tarayıcıların hesaplama, grafik işleme ve depolama konusunda sınırlamaları olduğundan, web tarayıcıları için de en uygunudur [28]. MobileNet mimarisi Şekil 4.13'de sunulmuştur.

Çizelge 4.1. MobileNet mimarisi [28]

Katman	Filtre	Giriş Boyutu
Evriştirme / s2	3x3x3x32	224x224x3
Evriştirme dw / s2	3x3x3x32 dw	112x112x32
Evriştirme / s1	1x1x32x64	112x112x32
Evriştirme dw / s2	3x3x3x64 dw	112x112x64
Evriştirme / s1	1x1x64x128	56x56x64
Evriştirme dw / s1	3x3x128 dw	56x56x128
Evriştirme / s1	1x1x128x128	56x56x128
Evriştirme dw / s2	3x3x128 dw	56x56x128
Evriştirme / s1	1x1x128x256	28x28x128
Evriştirme dw / s1	3x3x256 dw	28x28x256
Evriştirme / s1	1x1x256x256	28x28x256
Evriştirme dw / s2	3x3x256 dw	28x28x256
Evriştirme / s1	1x1x256x512	14x14x256
5x (Evriştirme dw / s1	3x3x512 dw	14x14x512
Evriştirme / s1)	1x1x512x512	14x14x512
Evriştirme dw / s2	3x3x512 dw	14x14x512
Evriştirme / s1	1x1x512x1024	7x7x512
Evriştirme dw / s2	3x3x1024 dw	7x7x1024
Evriştirme / s1	1x1x1024x1024	7x7x1024
Ortalama Havuzlama / s1	Havuzlama 7x7	7x7x1024
Tam Bağlantılı Katman / s1	1024x1000	1x1x1024
Softmax Fonksiyonu / s1	Sınıflandırıcı	1x1x1000

4.4.2. MobileNetV2

MobileNetV2, mobil ve gömülü cihazlarda verimli çalışabilen ve performans açısından daha iyi sonuçlar vermeyi amaçlayan evrişimsel sinir ağıdır. MobileNetV2 evrişimsel sinir ağı 53 katman ve 3,4 milyon parametreden oluşmaktadır. MobileNetV2 mimarisi 32 filtreli ilk tam evrişim katmanını ve ardından 19 artık darboğaz katmanını içerir. MobileNet temel mimarisi üzerine artık bağlantılar ve darboğaz katmanı eklenmiştir. Darboğaz artık bloğu katmanların arasına yerleştirilmiştir. MobileNet mimarisinde yer alan derinlemesine ayrılabilir evrişim yerine

darboğaz artık bloğu geliştirilmiştir. Darboğaz artık bloğu ağın aktivasyonlarını daha verimli bir şekilde hesaplamasına ve aktivasyondan sonra daha fazla bilgiyi korumasına izin verir. MobileNet mimarisinde bulunan noktasal evrişimler kanal sayısını sabit tutarken veya artırırken MobileNetV2 mimarisinde yer alan darboğaz artık blokları kanal sayısını azaltır. Darboğaz artık katmanları doğrusal bir yapıda olduğu için ayrıca doğrusal olmayan katmanların çok fazla bilgi kaybetmesini engeller [29]. MobileNetV2 genel mimarisi ana hatlarıyla Şekil 4.14'de sunulmuştur.

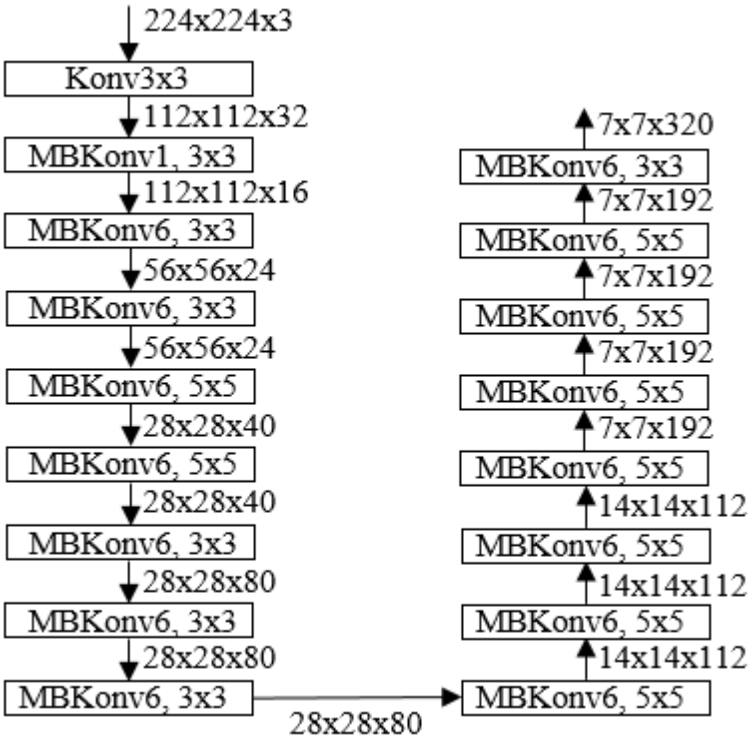
Çizelge 4.2. MobileNetV2 genel mimarisi [29]

Giriş	Operatör	t	c	n	s
224^2x3	konv2d	-	32	1	2
112^2x32	darboğaz	1	16	1	1
112^2x16	darboğaz	6	24	2	2
56^2x24	darboğaz	6	32	3	2
28^2x32	darboğaz	6	64	4	2
14^2x64	darboğaz	6	96	3	1
14^2x96	darboğaz	6	160	3	2
7^2x160	darboğaz	6	320	1	1
7^2x320	konv2d 1x1	-	1280	1	1
7^2x1280	orthvz 7x7	-	-	1	-
1x1x1280	konv2d 1x1	-	k	-	

4.4.3. EfficientNetB0

EfficientNet evrişimsel sinir ağı mimarisinde bileşik katsayı kullanan yeni bir ölçekleme modeli önerilmektedir. Genişlik, derinlik ve çözünürlük gibi ağ boyutlarını rastgele ölçekleyen evrişimsel sinir ağlarının aksine her ağ boyutu sabit bir ölçekleme katsayısı ile eşit olarak ölçeklenir. Bileşik ölçekleme yöntemi geleneksel ölçekleme yöntemlerine göre model doğruluğu ve verimliliğini artırmıştır. Bileşik ölçekleme yöntemi giriş görüntüsü büyükse daha fazla katmana ve büyük görüntüdeki daha küçük detayları tespit edebilmek için daha fazla kanala ihtiyaç duyulduğunu tespit edebilir. EfficientNet mimarisi temel olarak mobil ters çevrilmiş darboğaz evrişimini kullanır. EfficientNetB0, EfficientNet ağının ağının mobil ve gömülü cihazlar için revize edilmiş halidir. EfficientNetB0 ağı 5,3 milyon parametreden oluşmaktadır. EfficientNetB0 sıkma ve uyarlama (squeeze-and-excitation blocks) bloklarına ek olarak MobileNetV2 ağında kullanılan ters çevrilmiş

darboğaz artık bloklarını temel alır [30]. EfficientNet B0 mimarisi ana hatlarıyla Şekil 4.13'de sunulmuştur.



Şekil 4.13. EfficientNet B0 mimarisi [30]

4.4.4. NASNetMobile

NASNet, pekiştirmeli öğrenme yöntemini kullanarak en uygun evrişimsel sinir ağı mimarisinin oluşturulmasını sağlar. NASNet ağında kullanılan pekiştirmeli öğrenmede her arama işlemi sonucu ödül aranan mimarinin eğitilen veri seti üzerinde elde edilen doğruluktur. Farklı veri setleri üzerinde genellikle iyi sonuçlar veren evrişim katmanları pekiştirmeli öğrenme sayesinde yoğun olarak kullanılabilir. NASNet ağında genel mimari tanımlanmış olsa da evrişimli hücreler belli değildir. NASNet ağında yer alan normal ve indirgeme hücrelerinin yapıları denetleyici tekrarlayan sinir ağları tarafından aranır. NASNet ağında yer alan evrişimli hücrelerin tekrar sayısı ve ilk evrişim filtrelerinin sayısı ayarlanabilir parametrelerdir. Ayarlanabilir parametreler ölçeklendirme için kullanılır. Evrişimli hücreler normal ve indirgeme hücresi olmak üzere iki türdür. Normal hücre aynı boyutta özellik haritası döndüren evrişimli hücrelerdir. İndirgeme hücresi özellik haritasının yükseklik ve genişliğinin iki kat azaltıldığı hücrelerdir. NASNet Mobile, NASNet ağının mobil ve gömülü cihazlar için revize edilerek parametre sayısının düşürülmüş halidir. Orijinal NASNet ağı 88.9 milyon parametreden oluşurken NASNet Mobile 12 hücre ve 5.6 milyon parametre içermektedir [31].

4.5. Evrişimsel Sinir Ağı Yazılım Kütüphaneleri

Evrişimsel sinir ağı modellerin hazırlanması, eğitilmesi, test edilmesi ve kullanılabilmesi için çeşitli yazılım kütüphaneleri mevcuttur. Akademik çalışmalarda yaygın olarak kullanılan yazılım kütüphaneleri Çizelge 4.3'de görülmektedir.

Çizelge 4.3. Derin öğrenme yazılım kütüphaneleri

Kütüphane	Programlama Dili	Geliştirici Şirket veya Kişi
Tensorflow	Python	Google
Theano	Python	Montreal Üniversitesi
NumPy	Python	Travis Oliphant
Scikit-Learn	Python	David Cournapeau
DeepLearning4j	Java	Adam Gibson
Apache MXNet	C++ , Python , Java , Julia , MATLAB , JavaScript , Go , R , Scala , Perl ve Wolfram	Apache
Keras	Python	François Chollet
Pytorch	Python, C, CUDA	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan

4.5.1. Tensorflow

Tensorflow makine öğrenmesi ve yapay zeka çalışmalarında kullanılan Google tarafından C++ programlama dili kullanılarak geliştirilmiş ücretsiz ve açık kaynak yazılım kütüphanesidir. Tensorflow Javascript, C++, Java ve Python programlama dillerini desteklemektedir. Tensorflow ayrıca Linux, macOS, Windows, Android ve iOS işletim sistemlerini desteklemektedir [32]. Tensorflow hesaplama işlemleri için tensorları kullanır. Tensor, veri türlerini temsil eden n boyutlu bir vektörtür/matristir. Vektör tek boyutlu bir tensorken matris iki boyutlu tensordur. Tensorflow'da tüm işlemler graflar içerisinde gerçekleşir. Graf, modeldeki işlemleri temsil eden düğüm ve art arda gerçekleşen bir hesaplama kümesidir. Graf içerisinde tüm hesaplamalar tensorların birbirine bağlanmasıyla yapılır [33]. Tensorflow merkezi işlem birimi (CPU), grafik işlem birimi (GPU) ve tensor işlem birimi (TPU) üzerinde çalışabilir. Tensorflow kullanılarak yapılan çalışmaları görselleştirmek için Tensorboard uygulaması kullanılmaktadır [34]. Tensorflow kütüphanesi görüntü tanıma ve sınıflandırma problemlerinde daha iyi sonuçlar vermesinden ve literatürde yaygın olarak

kullanılmasından dolayı yapılan çalışmada tercih edilmiştir.

4.5.2. Theano

Montreal Üniversitesi Montreal Öğrenme Algoritmaları Enstitüsü tarafından geliştirilen açık kaynaklı yazılım kütüphanesidir. Theano evrişimsel sinir ağları tarafından öğrenilen büyük miktardaki verinin tekrarlı eğitim sürecinin verimli ve hızlı bir şekilde gerçekleşmesini sağlamaktadır. Theano, makine öğrenmesi algoritmalarında kullanılan çok boyutlu matematiksel ifadelerin tanımlanmasına, matematiksel ifadelerin optimize edilmesine ve grafiksel işlem birimi (Graphics Processing Unit) kullanımına olanak vermektedir [35].

4.5.3. NumPy

NumPy, Python progromlama dili için için çok boyutlu dizileri ve matrisleri destekleyen matematiksel kütüphanedir. Derin öğrenmede tercih edilmesinin nedeni hesaplamalar için çok sayıda üst düzey matematiksel işlevleri desteklemesidir. NumPy temel olarak NumPy dizilerine dayanır. NumPy dizileri performans açısında hızlı ve kullanımı kolaydır. Bunun yanında diğer Python listelerine göre daha az depolama alanı kaplar. Yaygın olarak kullanılan bilgisayarlı görü kütüphanesi OpenCV Python verilerini depolamak ve üzerinde çalışmak için Numpy kütüphanesini kullanmaktadır [36].

4.5.4. Scikit-Learn

Scikit-Learn, Python programlama dili için geliştirilmiş makine öğrenmesi yazılım kütüphanesidir. Scikit-Learn büyük ölçüde NumPy, SciPy ve Matplotlib kütüphaneleri üzerine geliştirilmiştir. Destek vektör makineleri, rastegele ormanlar, gradyan artırma ve k-means olmak üzere çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarını içermektedir. Scikit-Learn, NumPy ve diper Python matematiksel kütüphaneleri ile birlikte çalışabilecek şekilde tasarlanmıştır. Scikit-Learn, veri sınıflandırma, veri modelleme ve veri ön işleme için de kullanılmaktadır. Ayrıca verilerdeki özniteliklerin sayısını azaltmak için de kullanılabilir [37].

4.5.5. DeepLearning4j

DeepLearning4j, Java programlama dili için geliştirilmiş derin öğrenme algoritmalarının çoğunu destekleyen yazılım kütüphanesidir. DeepLearning4j evrişimsel sinir ağı, tekrarlayan sinir ağı ve uzun kısa süreli bellek gibi farklı sinir ağlarını destekler. Hadoop ve Apache Spark ile entegre edilmiştir. Yapay sinir ağları Hadoop ve Spark üzerinde dağıtık olarak eğitilebilir. Deeplearning4j, NumPy'nin Python'a sağladığı işlevlere benzer şekilde Java ve Scala'da bilimsel hesaplamaya izin veren ND4J kullanan n boyutlu bir dizi sınıfı içerir. Deeplearning4j CPU ve GPU üzerinde kullanılabilir [38].

4.5.6. Apache mxnet

Apache MXNeT, derin sinir ağlarını eğitmek ve dağıtmak için kullanılan yazılım çerçevesidir. Apache MXNeT ölçeklenebilir mimariye sahiptir ve modellerin daha hızlı eğitilmesini sağlamaktadır. Özellikle GPU'lar üzerinde yüksek hesaplama hızlarına ulaşmaktadır. C++ , Python , Java , Julia , MATLAB , JavaScript , Go , R , Scala , Perl ve Wolfram programlama dillerini desteklemektedir. Evrişimsel sinir ağlarını ve uzun kısa süreli bellek ağlarını desteklemektedir. Ayrıca Apache MXNeT eğitilen modellerin düşük donanım kaynaklarına sahip mobil cihazlara dağıtımını desteklemektedir [39].

4.5.7. Keras

Keras, yapay sinir ağları için geliştirilmiş Python yazılım kütüphanesidir. Keras Tensorflow, Theano ve CNTK için bir arayüz sunmaktadır. 2.4 sürümünden itibaren yalnızca Tensorflow desteklenmektedir. Keras, derin sinir ağlarında hızlı denemeler yapmak için tasarlanmış olup kullanıcı dostu, modüler ve genişletilebilir bir mimariye sahiptir. Standart sinir ağlarına ek olarak evrişimsel sinir ağları ve tekrarlayan sinir ağları desteklenmektedir. Keras, kullanıcıların mobil cihazlarda, web'de veya java sanal makinesinde modeller eğitmesine olanak tanır. Ayrıca grafik işlem birimleri (GPU) ve tensör işlem birimleri (TPU) donanımlarında derin öğrenme modellerinin dağıtık olarak eğitilmesine izin verir [40].

4.5.8. Pytorch

PyTorch, Python ve Torch tabanlı derin öğrenme kütüphanesidir. Lua programlama dilinin düşük popülerliğinden dolayı, Torch, Google'ın TensorFlow'un sahip olduğu büyümeyi gösteremedi. Bu nedenle Torch kütphanesinin temel alındığı Python programlama diline yönelik PyTorch kütüphanesi geliştirildi. PyTorch, çok boyutlu diziler üzerinde çalışılabilmesini sağlamak için tensorları kullanır. Tensor, birden fazla boyutta veri tutabilen veri birimidir. Numpy dizileri gibi bir sayı, vektör, matris veya çok boyutlu dizi olabilir. Tensorlar, Numpy dizileri ile benzer bir yapıya sahiptir. Tensorlar hem CPU hem de GPU tarafından kullanılabilir. Float Tensor, çift tensor, yarım tensor, integer tensor ve uzun tensor gibi çeşitli tensor türleri vardır. PyTorch, varsayılan olarak 32-bit float tensoru kullanır. PyTorch sinir ağlarını temsil etmek için otograd, optim ve nn modüllerini kullanır. Otograd modülü, PyTorch'un ileri geçişteki gradyanları hızlı bir şekilde hesaplamasına yardımcı olan otomatik farklılaşma motorudur. Optim modülü, sinir ağları oluşturmak için kullanılan optimize ediciler için önceden yazılmış algoritmalara sahip pakettir. Nn modülü, sinir ağı modellerinin oluşturulmasına yardımcı olan çeşitli sınıflar içerir [41].

5. TOPLULUK ÖĞRENME

Hansen ve Salamon [42] ilk kez bir topluluk sisteminin varyans azaltma özelliğini bir sinir ağının genelleme performansının, benzer şekilde yapılandırılmış sinir ağlarının bir topluluğu kullanılarak iyileştirilebileceğini gösterdi. Schapire topluluk sistemlerini makine öğrenme algoritmaları ile ilişkilendirmiş ve güçlendirme adını verdiği bir prosedür aracılığıyla zayıf sınıflandırıcıları birleştirerek daha güçlü bir sınıflandırıcının üretilebileceğini kanıtlamıştır [42].

Topluluk öğrenme, birden çok modelden gelen tahminleri birleştirerek daha iyi performans elde edilmesini amaçlayan öğrenme yöntemidir. Topluluk öğrenmede nihai tahmin birkaç modelden gelen sonuçların birleştirilmesiyle elde edilir. Modeller ne kadar çeşitli olursa topluluk tahmini o kadar güçlü olur. Topluluk öğrenme belirli bir sınıflandırma problemi için en uygun tek bir modelin aranması yerine birden fazla modelin birlikte kullanılmasını amaçlamaktadır. Topluluk öğrenme yöntemi eğitim verisinin fazla veya eksik olması durumunda da fayda sağlamaktadır. Eğitim verisi tek bir sınıflandırıcı eğitimini zorlaştıracak kadar büyük olduğunda veriler alt kümelere bölünür. Her bölüm uygun bir kombinasyon kuralı kullanılarak birleştirilebilen ayrı bir sınıflandırıcıyı eğitmek için kullanılır [42]. Eğitim verisi sayısının az olduğu durumda ise verilerin farklı önyükleme örnekleri farklı sınıflandırıcıları eğitmek için kullanılır. Topluluk modellerinin her biri aynı doğrulukta tahmin üretemeyebilir. Bu durumda modellerin tahminlerine farklı ağırlıklar verilerek doğru tahmin yapan modelin etkisi artırılabilir. Topluluk öğrenme yöntemi kullanılan sınıflandırıcı problem için yeterli olmadığı durumlarda da tercih edilebilir. Örneğin bir lineer sınıflandırıcı ile problem parabolik (polinom) olarak çözülmeye çalışılırsa lineer sınıflandırıcı yeterli olmayacaktır. Çoklu lineer sınıflandırıcıdan oluşan bir topluluk modeli parabolik çözümler üretebilir. Topluluk öğrenme yönteminin en önemli nedeni sınıflandırma başarısını artırmak için veri birleştirilmesidir. Aynı sınıf kümesine ait farklı veri dağılımları üzerinde eğitilmiş modeller daha iyi sonuçlar için tahmin süreci boyunca kullanılır [43].

Topluluk sisteminin başarısı topluluğu oluşturan sınıflandırıcı çeşitliliğine bağlıdır. Topluluk sistemindeki sınıflandırıcı çeşitliliği birkaç yöntemle elde edilebilir. Bu yöntemlerden ilki sınıflandırıcıları eğitmek için farklı eğitim veri kümeleri kullanmaktır. Farklı eğitim veri kümeleri genellikle ana eğitim veri kümesinden rastgele örnekler seçilerek elde edilir. Çeşitliliği elde etmek için diğer yöntem farklı sınıflandırıcılar için farklı eğitim parametrelerinin kullanılmasıdır. Örneğin bir dizi çeşitli sinir ağı farklı ağırlıklar, farklı katman sayıları, farklı öğrenme oranları vb. hiper parametreler değiştirilerek elde edilebilir. Elde edilen sinir ağları karar ağaçları, en yakın komşu sınıflandırıcı veya destek vektör makineleri kullanılarak birleştirilebilir [43].

Maksimum oylama, ortalama alma, ağırlıklı ortalama, uzmanların karışımı, torbalama, artırma, karıştırma ve yığınlama en yaygın topluluk öğrenme yöntemleridir [44].

5.1. Maksimum Oylama

Oylama modellerin üzerinde oy kullanabilecekleri farklı seçenekler olduğunda kullanılabilir. Maksimum oylama yönteminde en çok oyu alan seçenek seçilmiş olur. Maksimum oylama yöntemi genellikle sınıflandırma problemlerinde kullanılır. Toplulukta bulunan her makine öğrenme modeli oy kullanır ve en fazla oy alan seçenek tercih edilir. Sert ve yumuşak olmak üzere iki tür maksimum oylama vardır. Kesin oylamada sınıflandırıcı modeller oransız olarak bir tercihte bulunur ve çoğunluk hangi seçeneği tercih ettiyse o seçenek tahmin değeridir [43]. Örnek olarak 3 sınıflandırıcının sert oylama ile yaptığı tercihler ve seçim işleminin nasıl gerçekleştiği açıklanmıştır.

Sınıflandırıcı 1, Sınıf A'yı tahmin eder.

Sınıflandırıcı 2, Sınıf B'yi tahmin eder.

Sınıflandırıcı 3, Sınıf B'yi tahmin eder.

3 oydan 2'sini alan Sınıf B topluluk kararı olarak seçilir. 3 sınıflandırıcının yumuşak oylama ile yaptığı tercihler ve seçim işleminin nasıl gerçekleştiği açıklanmıştır.

Sınıflandırıcı 1, %99 olasılıkla A sınıfını tahmin eder.

Sınıflandırıcı 2, %49 olasılıkla A sınıfını tahmin eder.

Sınıflandırıcı 3, %49 olasılıkla A sınıfını tahmin eder.

A sınıfına ait olma olasılığı = (99 + 49 + 49)) / 3 = %65,67. Sonuç olarak topluluk kararı A sınıfıdır.

5.2. Ortalama Alma

Tahmin edilmeye çalışan veri üzerinde çeşitli modeller tarafından tahminler yapılır. Ortalama almada nihai çıktı tüm tahminlerin ortalamasıdır. Ortalama alma genellikle regresyon problemleri için kullanılmaktadır. Rastgele orman regresyonunda nihai sonuç bireysel karar ağaçlarından elde edilen tahminlerin ortalamasıdır [43]. Örnek olarak bir emtiyanın fiyatının aşağıdaki şekilde tahmin eden üç regresyon modelinin tahminleri aşağıdaki gibidir.

Regresör 1, 200 olarak tahmin yapar.

Regresör 2, 300 olarak tahmin yapar.

Regresör 3, 400 olarak tahmin yapar.

Topluluk kararı 3 modelin ortalaması ((200 + 300 + 400)/3) olan 300 değerine eşittir.

5.3. Ağırlıklı Ortalama

Ağırlıklı ortalama yönteminde birden çok model arasında her bir modelin ne kadar iyi olduğuna bağlı olarak tahmine katkıda bulunmasına olanak verir. Bir model genel olarak veri kümesinde daha iyi performans gösteriyorsa ona daha yüksek ağırlık verilir. Bu sayede önyargı azaltılır ve genel performans iyileştirilir [43]. Örnek olarak bir emtiyanın fiyatının aşağıdaki şekilde tahmin eden üç regresyon modelinin tahminleri ve ağırlıkları aşağıdaki gibidir.

Regresör 1, 200 olarak tahmin yapar ve ağırlığı 0,35'dir.

Regresör 2, 300 olarak tahmin yapar ve ağırlığı 0,45'dir.

Regresör 3, 400 olarak tahmin yapar ve ağırlığı 0,20'dir.

Topluluk kararı 3 modelin ağırlıklı ortalaması ((0,35*200 + 0,45* 300 + 0,20*400)/3) olan 285 değerine eşittir.

5.4. Uzmanların Karışımı

Farklı sınıflar üzerine eğitilen iki modelin birlikte kullanılmasını sağlayan topluluk öğrenme yöntemidir. Uzmanların karışımı topluluk öğrenme yöntemi tahmine dayalı modelleme görevlerini alt görevlere ayırmayı, her biri için bir uzman modeli eğitmeyi, tahmin edilecek girdiye dayalı olarak hangi uzmana güvenileceğini öğrenen ve tahminleri birleştiren bir yönlendirici modeli geliştirmeyi içerir. Bu şekilde eğitilen modellerin ağırlıkları farklı bir sinir ağı tarafından tekrar eğitilir. Eğitilen sinir ağı bu ağırlıklara göre veriler üzerinde tahmin yapar. Uzmanların karışımı yönteminde ilk olarak bir görev alt görevlere ayrılır. Her alt görev için bir uzman model geliştirilir. Hangi uzmanın kullanılacağına karar vermek ve tahminleri birleştirmek için yeni bir yönlendirici model geliştirilir ve uygulanır [44]. Tahmin yapmak için yönlendirici model çıktısı kullanılır.

5.5. Torbalama

Torbalama, eğitim verilerinin alt örneklere ayrılarak çeşitli modellerin her bir alt örnek ile eğitildiği ve tahminin eğitilen modellerin tahminlerinin birleştirilmesiyle elde edildiği topluluk öğrenme yöntemidir. Veri setinde yer alan aynı veriler birden fazla alt örnekte bulunabilir. Topluluk üyeleri tarafından yapılan tahminler daha sonra oylama veya ortalama alma gibi teknikler kullanılarak birleştirilir [45]. Torbalama yönteminin en önemli noktası veri kümesinin her örneğinin hazırlanma şeklidir. Torbalama yöntemi genişletilebilir bir yapıya sahiptir. Örneğin eğitim veri setinde daha fazla

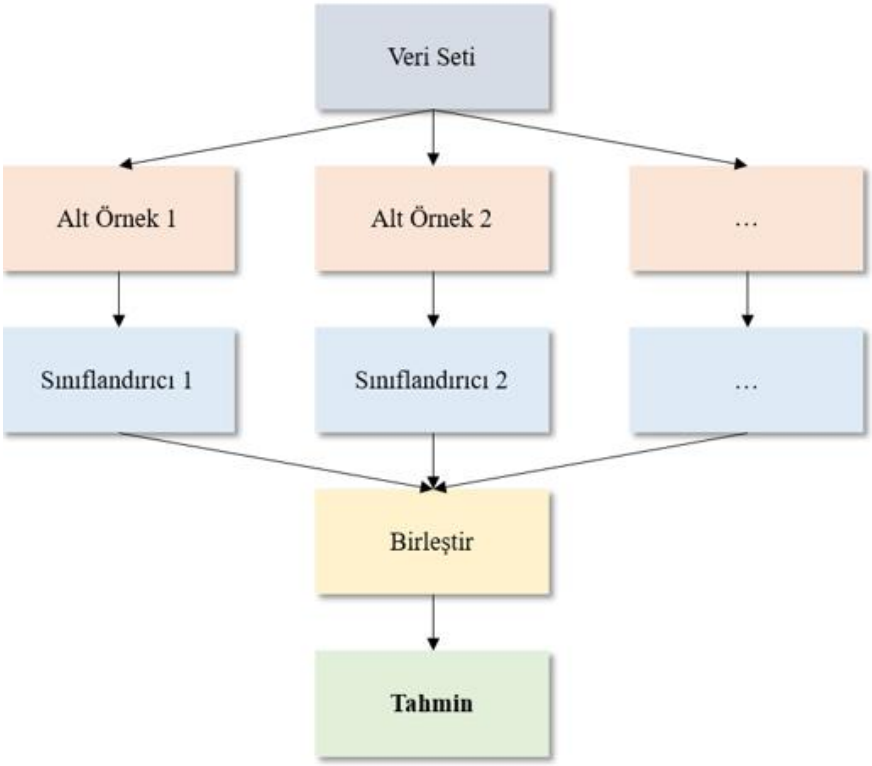
değişiklik yapılabilir, eğitim verisine uyan algoritma veya tahmin için kullanılan mekanizma değiştirilebilir. Torbalama yönteminin temel amacı topluluk tahminlerindeki varyansı azaltmaktır. Torbalama topluluk öğrenme yöntemine dayalı popüler yöntemler aşağıdaki gibidir.

- Torbalı Karar Ağaçları

- Rastgele Orman Sınıflandırıcıları

- Ekstra Ağaçlar

Torbalama topluluk öğrenme yönteminin çalışma mekanizması Şekil 5.1'de görülmektedir.



Şekil 5.1. Torbalama topluluk öğrenme yöntemi çalışma mekanizması

5.6. Artırma

Artırma, daha iyi tahminler yapabilmek için önceki tahmin hatalarından öğrenen topluluk öğrenme yöntemidir. Artırma topluluk öğrenme yönteminin temel özelliği tahmin hatalarını düzeltme fikridir. Artırma topluluk öğrenme yöntemi birkaç zayıf temel modeli tek bir güçlü model oluşturmak için birleştirir. Zayıf modeller daha iyi tahminler oluşturmak için sonraki modelden öğrenir. Zayıf modellerin tahminleri basit oylama veya ortalama alma kullanılarak birleştirilir. Modellerin katkıları performansları ile orantılı olarak ağırlıklandırılır. Bu sayede zayıf modeller ile güçlü tahminler yapabilmektir [44]. Artırma yönteminde veri seti sıralı olarak işlenir. İlk sınıflandırıcı tüm veri seti ile beslenir ve tahminler analiz edilir. Sınıflandırıcı 1'in doğru tahminler üretemediği durumlar ikinci

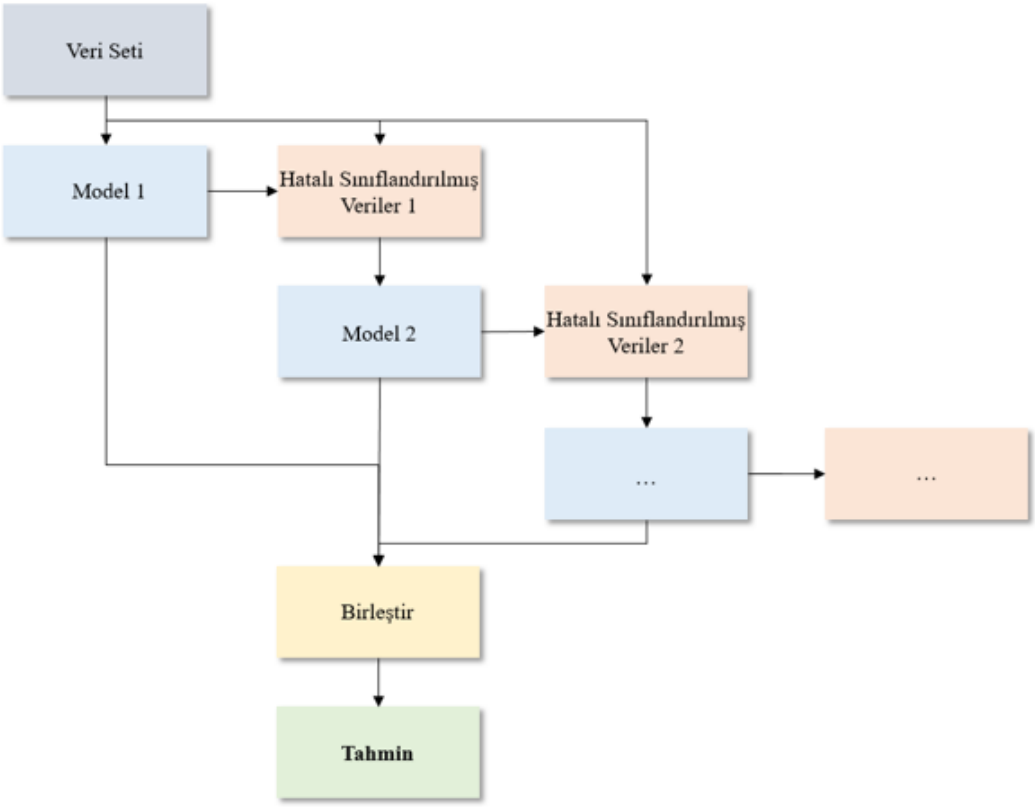
sınıflandırıcıya gönderilir. Bu işlem Sınıflandırıcı 2'nin sorunlu alanlara odaklanabilmesi ve uygun karar verebilmesi için yapılır. Yanlış tahminlere daha fazla ağırlık verilir ve son modeldeki hataları düzelten modeller eklenir. Tüm modeller aynı veri seti ile eğitilirken hatalı durumlar sıralı olarak tekrar öğrenilir. Son olarak tüm modellerin ortalaması kullanılarak nihai model elde edilir [45]. Artırma topluluk öğrenme yöntemine dayalı popüler yöntemler aşağıdaki gibidir.

- Uyarlanabilir Güçlendirme (AdaBoost)

- Stokastik Gradyan Yükseltme (XGBoost ve benzeri)

- Gradyan Yükseltme Makineleri

Artırma topluluk öğrenme yönteminin çalışma mekanizması Şekil 5.2'de görülmektedir.



Şekil 5.2. Artırma topluluk öğrenme yöntemi çalışma mekanizması

5.7. Yığınlama

Eğitim verilerini farklı model türleri ile eğitip tahminleri birleştirmek için ayrı bir model kullanan en yaygın topluluk öğrenme yöntemidir. Yığınlama topluluk öğrenme yönteminde ilk olarak topluluktaki tüm modeller mevcut veri ile eğitilir. Ardından diğer algoritmaların tüm tahminleri ek girdiler olarak kullanılarak nihai bir tahmin yapabilmek için birleştirici meta model eğitilir. Birleştirici meta modeli olarak genellikle lojistik regresyon modeli kullanılır. Lojistik regresyon bir veya daha fazla bağımsız

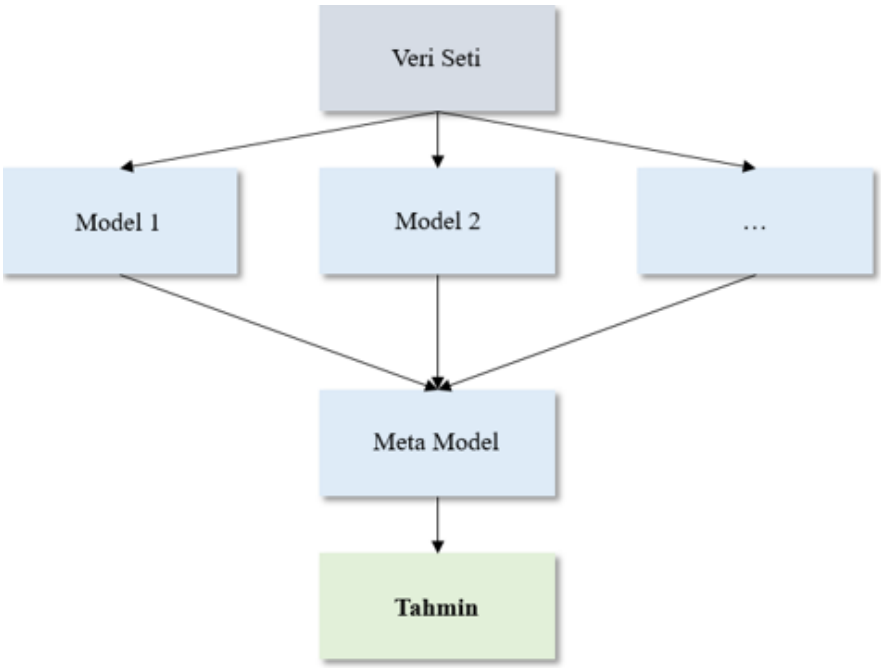
değişkenin doğrusal bir kombinasyonu olmasını sağlayarak gerçekleşen olayın olasılığını modelleyen istatiksel bir modeldir. Birleştirici meta model toplulukta bulunan eğitilmiş modellerden daha iyi performans gösterir. Yığınlama topluluk üyeleri seviye-0 modelleri olarak adlandırılırken tahminleri birleştirirken kullanılan model seviye-1 model olarak adlandırılmaktadır. Daha fazla model katmanı kullanılabilmesine rağmen iki seviyeli model hiyerarşisi en yaygın kullanılan yaklaşımdır. Yığınlama yönteminde her sınıflandırıcı modelden gelen tahminler bir araya toplanır ve tahmin edici meta modele gönderilir. Yığınlama hem regresyon hem de sınıflandırma problemleri için kullanılabilir. Regresyon problemlerinde meta modele iletilen değerler sayısaldır. Sınıflandırma problemlerinde meta modele iletilen değerler olasılıklar veya sınıf etiketleridir [45]. Yığınlama topluluk öğrenme yöntemine dayalı popüler yöntemler aşağıdaki gibidir.

- Yığılmış Modeller

- Karıştırma

- Süper Topluluk

Yığınlama topluluk öğrenme yönteminin çalışma mekanizması Şekil 5.3'de görülmektedir.



Şekil 5.3. Yığınlama topluluk öğrenme yöntemi çalışma mekanizması

5.8. Karıştırma

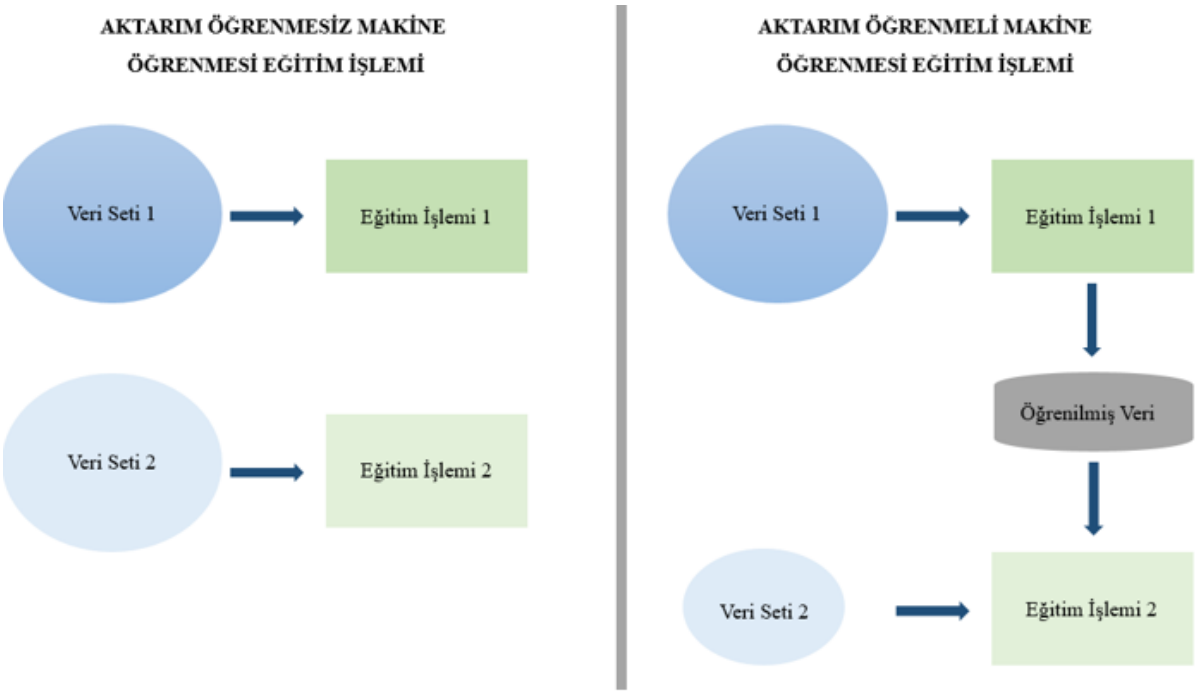
Karıştırma, yığınlama topluluk öğrenme yöntemi gibi farklı modellerin tahminlerinin birleştirildiği meta model kullanır. Yığınlama yönteminden farklı olarak tahmin yapmak için eğitim seti kullanılarak hazırlanan bir uzatma seti kullanır. Karıştırma, meta modelin uzatma doğrulama seti

üzerinde topluluk üyesi modeller tarafından yapılan tahminler üzerine eğitildiği bir yığınlama türüdür. Topluluk üyeleri modeller tarafından yapılan tahminler uzatma seti kullanılarak yapılır. Tahminler ve uzatma seti test verisi üzerinde nihai tahminler yapan meta modelin eğitilmesi için kullanılır. Bu durumda meta model ile topluluk üyesi modeller farklı veri seti üzerinde eğitilmiş olur. Meta verinin eğitimi için daha az veri kullanılması sağlanır [44].

6. AKTARIM ÖĞRENME

Aktarım öğrenme, halihazırda eğitilmiş makine öğrenmesi modelinin ilgili farklı bir soruna uygulanmasıdır. Aktarım öğrenmede amaç bir modelin öğrendiği eğitim verisinin başka bir görev için daha az veri ile eğitilerek kullanılabilmesini sağlamaktır. Aktarım öğrenme modelleri bir problemi çözerken kazanılan bilgiyi depolamaya ve onu farklı fakat ilgili bir probleme uygulamaya odaklanır. Aktarım öğrenme hedef göreve benzer bir kaynak model seçmeyi, bilgiyi aktarmadan önce kaynak modeli hedef modele uyarlamayı ve hedef modele ulaşmak için kaynak modeli eğitmeyi içerir [46]. Kaynak görevden hedef göreve aktarılan temel bilgiler aynı olduğundan modelin son katmanlarında ince ayar yapmak ve ilk ve orta katmanları dondurmak yaygın bir uygulamadır. Az miktarda veri içeren görevlerde kaynak model hedef modele çok benziyorsa aşırı uyum sorunu olabilir. Aktarım öğrenme modelinin aşırı uyumunu önlemek için öğrenme oranını ayarlamak, kaynak modelden bazı katmanları dondurmak veya hedef modeli eğitirken doğrusal sınıflandırıcılar eklemek aşırı uyum sorununun önlenmesine yardımcı olabilir. Aktarım öğrenme sayesinde öğrenme işlemi öğrenilmiş ağırlıklar üzerinden başlar. Aktarım öğrenme sayesinde eğitim süresi kısalır, daha düşük kaynaklar kullanılır, daha az veriye ihtiyaç duyulur ve daha yüksek performans elde edilir.

Aktarım öğrenme çok büyük miktarda hesaplama gücü gerektirmeleri nedeniyle çoğunlukla bilgisayarla görme, ses tanıma ve doğal dil işleme görevlerinde kullanılır. Yapay sinir ağları genellikle ilk katmanlarda kenarları, orta katmanda şekilleri ve son katmanlarda göreve özgü özellikleri algılamaya çalışır. Aktarım öğrenmede önceden eğitilen modelin ilk ve orta katmanları aynen kullanılırken son katmanları yeniden eğitilir. Aktarım öğrenmede modelin üzerinde eğildiği önceki görevden yeni göreve mümkün olduğu kadar fazla bilgi aktarılmaya çalışılır. Aktarılacak bilgi probleme göre çeşitli şekillerde olabilir. Aktarım yapılan model çözülmeye çalışılan sorunla daha az ilgili ise performans düşebilir. Bu durum negatif aktarım olarak adlandırılır [47]. Aktarım öğrenme kullanılarak yapılan makine öğrenmesi eğitim işlemi aktarım öğrenme kullanılmadan yapılan eğitim işleminin farkı Şekil 6.1'de görülmektedir.



Şekil 6.1. Aktarım öğrenme kullanılmadan ve kullanılarak yapılan makine öğrenmesi eğitim işlemi [47]

Şekil 6.1'de görüldüğü gibi aktarım öğrenme ile önceden öğrenilmiş veriler sayesinde daha az veri ile öğrenme işlemi gerçekleştirilebilir. Aktarım öğrenme temel olarak aşağıdaki 3 faydayı sağlar.

- Aktarım öğrenmede önceden eğitilmiş bir model kullanmak sıfır olarak başlamaktan daha iyi bir temel sağlar. Bazı özellikler eğitim almadan bile kazanılabilir.
- Model önceden benzer bir görev için eğitilmiş olması nedeniyle daha yüksek öğrenme oranına sahiptir.
- İyi bir başlangıç temeli ve daha yüksek öğrenme oranı sayesinde model daha yüksek performansta çalışabilir.

Aktarım öğrenme kullanılarak örnek aktarımı, özellik temsili aktarımı, parametre aktarımı ve ilişkisel bilgi aktarımı yapılabilir [47].

Örnek aktarımı, kaynak görevde öğrenilen bilginin hedef görevde yeniden kullanılmasıdır. Çoğu durumda kaynak görevde öğrenilen bilgi doğrudan yeniden kullanılamaz.

Özellik temsili aktarımı, kaynak görevde öğrenilen özelliklerin hedef görevde kullanılmasıdır. Özellik temsiline dayalı aktarımlar için denetimli veya denetimsiz yöntemler kullanılabilir.

Parametre aktarımı, hedef görev için önceden eğitilmiş modelin bazı parametrelerin veya hiper parametrelerin paylaşıldığı yöntemdir.

İlişkisel bilgi aktarımı, her bir veri noktasından diğer veri noktaları ile ilişkili olanlarının aktarılmasıdır.

Alanlar arası benzerlik ve verinin etiketli veya etiketsiz olma durumuna göre tümevarımlı aktarım öğrenme, denetimsiz aktarım öğrenme, transdüktif aktarım öğrenme olmak üzere 3 tür aktarım öğrenme vardır [48].

6.1. Tümevarımlı Aktarım Öğrenme

Tümevarımlı aktarım öğrenme yönteminde modelin üzerinde çalıştığı belirli görevler farklı olsa da kaynak ve hedef alanlar aynıdır. Tümevarımlı aktarım öğrenmede kaynak modeldeki bilgi hedef görevi geliştirmek için kullanılmaya çalışılır. Model, hedef görevin performansını iyileştirmeye yardımcı olmak için kaynak görevdeki tümevarımsal sapmaları kullanır. Tümevarımlı aktarım öğrenme, veri setinin etiketli veri içerip içermemesine bağlı olarak iki alt kategoriye ayrılır. Bunlar sırasıyla çok görevli öğrenmeyi ve kendi kendine öğrenmeyi içerir [48].

6.2. Denetimsiz Aktarım Öğrenme

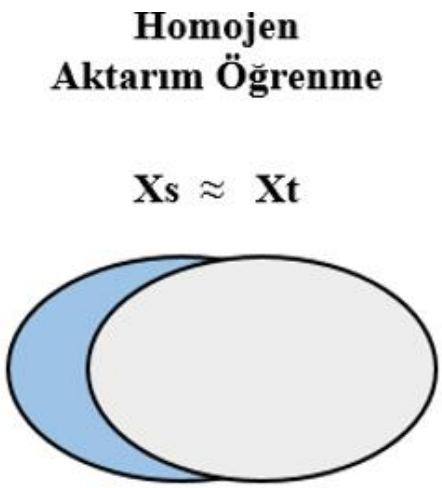
Denetimsiz öğrenme etiketlenmemiş veriler üzerinde öğrenme işleminin gerçekleştirilmesidir. Denetimsiz öğrenme algoritması, veri kümesindeki kalıpları tanımlar ve öğrenir. Denetimsiz aktarım öğrenme etiketsiz veriler üzerinde eğitilen modelin tekrar kullanılmasıdır. Hem kaynak hem hedef görevlerde etiketlenmemiş veri kümesinin kullanıldığı aktarım öğrenme türüdür [48].

6.3. Transdüktif Aktarım Öğrenme

Kaynak ve hedef görevlerin etki alanlarının tam olarak aynı olmadığı ancak ilişkili olduğu durumlarda kullanılan aktarım öğrenme türüdür. Bu aktarım öğrenme senaryosunda genellikle kaynak etki alanında çok sayıda etiketli veri bulunurken hedef etki alanında yalnızca etiketlenmemiş veriler bulunur [48]. Etiketli veya etiketsiz verilerin bulunma durumuna bakmaksızın yalnızca alanların benzerliğine göre homojen aktarım öğrenme ve heterojen aktarım öğrenme olmak üzere iki ana sınıfa ayrılır [49].

6.4. Homojen Aktarım Öğrenme

Homojen aktarım öğrenmede kaynak görev ile hedef görev özellik uzayının yakın olması sağlanır. Bunun için kaynak ve hedefteki marjinal ve koşullu dağılım farklılıkları düzeltilir [49]. Homojen aktarım öğrenmede kaynak ve hedef özellik uzayının gösterimi Şekil 6.2'de görülmektedir.

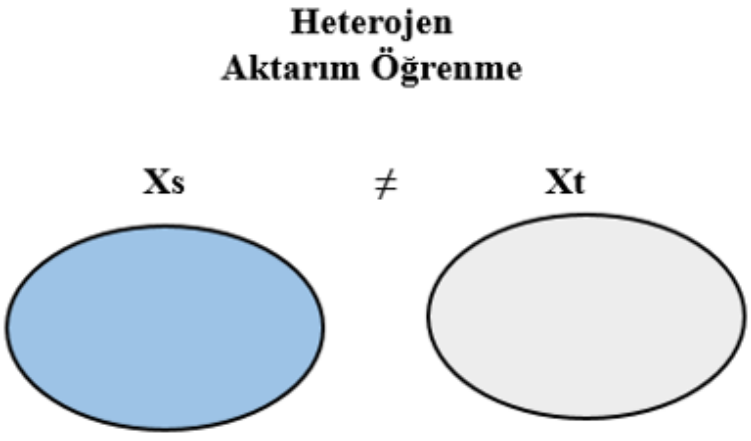


Şekil 6.2. Homojen aktarım öğrenme kaynak ve hedef özellik kümeleri

Şekil 6.2'de görüldüğü gibi Xs kaynak özellik kümesini temsil ederken Xt hedef özellik uzayını temsil etmektedir.

6.5. Heterojen Aktarım Öğrenme

Heterojen aktarım öğrenmede kaynak ve hedef etki alanları hiçbir özelliği veya etiketi paylaşamayabilir. Bu nedenle heterojen aktarım öğrenmede kaynak ve hedef farklı özellik uzaylarına sahiptir. Heterojen aktarım öğrenmede kaynak ve hedef özellik uzayları arasında boşluğu gidermek için daha fazlam marjinal ve koşullu dağıtım yapılarak sorun homojen aktarım öğrenme problemine indirgenir [50]. Heterojen aktarım öğrenmede kaynak ve hedef özellik uzayının gösterimi Şekil 6.3'de görülmektedir.



Şekil 6.3. Heterojen aktarım öğrenme kaynak ve hedef özellik kümeleri

Şekil 6.3'de görüldüğü gibi Xs kaynak özellik kümesini temsil ederken Xt hedef özellik uzayını temsil etmektedir.

6.6. Aktarım Öğrenme Uygulama Adımları

Adım 1. Göreve uygun önceden eğitilmiş model belirlenir. Aktarım öğrenmenin faydalı olması için önceden eğitilmiş model ile hedef görev alanı arasında ilişki olmalıdır.

Adım 2. Önceden eğitilmiş ağırlıklarla temel bir model oluşturulur. Önceden eğitilmiş model ağırlıklarına ResNet veya Xception gibi mimariler aracılığıyla erişilebilir. Temel modelin son çıktı katmanında kullanım durumunda ihtiyaç duyulmayan fazla nöronlar bulunabilir. Bu senaryolarda son çıktı katmanı kaldırılarak yeni son çıktı katmanı eklenir.

Adım 3. Ağırlıkları yeniden başlatmayı azaltmak için önceden eğitilmiş model katmanları dondurulur. Özellikle ilk katmanların dondurulması öğrenilmiş ağırlıkları korumak açısından önemlidir. Katmanların dondurulması sayesinde halihazırda öğrenilen bilgiler kullanılabilecektir.

Adım 4. Dondurulmuş katmanların üzerine yeni katmanlar eklenir. Bu katmanlar genellikle çıktı katmanlarıdır.

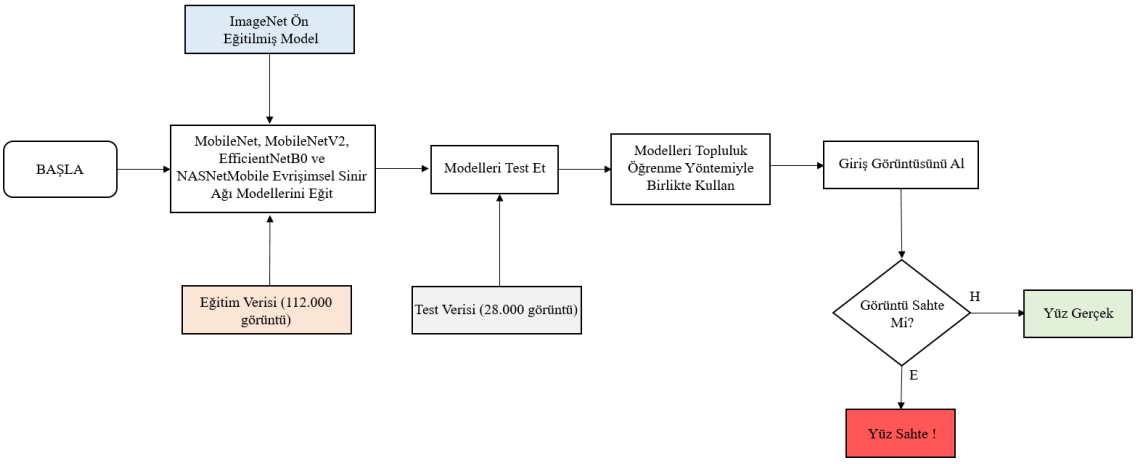
Adım 5. Eklenen katmanların hedeflenen görev için tekrar eğitilmesi sağlanır.

Adım 6. Model performansının artırılması için ayarlanması sağlanır. Modelin düşük öğrenme oranı ile eğitilmesi performansı artırırken aşırı uyumu da engeller.

7. ARAŞTIRMA BULGULARI

7.1. Uygulamanın Yapılışı

Literatürde yapılan çalışmalarda sahte yüz görüntüsü tespitinde klasik görüntü işleme teknikleri yerine daha yüksek doğruluk oranı sağlayan derin öğrenme algoritmaları kullanılmaktadır. Sahte yüz tespitinin yüksek doğrulukta yapılabilmesinin yanında kaynakların verimli kullanılması gerekmektedir. Nesnelerin interneti milyarlarca cihazın birbirine bağlanması işlemlerin merkezi sunucular üzerinde yapılabilmesini giderek zorlaştıracaktır. Bunun için geliştirilecek sahte yüz tespiti modeli mobil cihazlarda çalışabilmelidir. Bu nedenle yapılan çalışmada mobil cihazlarda maksimum performansta çalışabilecek hafif evrişimsel sinir ağı modelleri kullanılmıştır. Yapılan çalışmanın akış diyagramı Şekil 7.1'de görülmektedir.



Şekil 7.1. Akış diyagramı

Şekil 7.1'de sahte yüz tespiti modelinin geliştirilmesi için yapılan çalışmalar görülmektedir. Sahte yüz tespiti modelini eğitmek için 70.000 gerçek ve 70.000 sahte görüntü içeren veri seti kullanılmıştır. Veri setinde yer alan 112,000 görüntü eğitim ve 28,000 görüntü test işlemi için kullanılmıştır. Eğitim işlemi için MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile hafif evrişimsel sinir ağları kullanılmıştır. İlk olarak eğitim işlemi için MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile evrişimsel sinir ağları ayrı ayrı eğitilmiştir. Eğitim işleminde modeller ImageNet üzerinde ön eğitilmiş olarak transfer öğrenme ile tekrar kullanılmıştır. İlk eğitimler sonucunda EfficientNetB0 algoritması %93,64 ile en yüksek doğruluk oranına ulaşmıştır. Doğruluk oranını artırabilmek için en yüksek EfficientNetB0 algoritmasına transfer öğrenme için ReLU aktivasyon fonksiyonuna sahip iki yoğun katman (256 nöron), iki bırakma katmanı, bir düzleştirme katmanı, ReLU aktivasyon fonksiyonuna sahip bir yoğun katman (128) ve sınıflandırma için kullanılan softmax aktivasyon fonksiyonuna sahip iki düğümlü yoğun katman eklenmiştir. Bu işlem sonucu %95,48 doğruluk oranına ulaşılmıştır.

Son olarak %95,48 doğruluk oranına ulaşan model ile MobileNet ve MobileNetV2 modelleri yığınlama topluluk öğrenme yöntemi ile birlikte eğitilerek %96,44 ile en yüksek doğruluk oranına ulaşılmıştır.

7.2. Veri Seti

Evrişimsel sinir ağı modelini eğitmek için 140.000 görüntüden oluşan veri seti kullanılmıştır. Kullanılan veri seti FFHQ veri setindeki 70.000 gerçek ve StyleGAN2 ile üretilmiş 70.000 sahte yüz görüntüsü içermektedir [18]. FFHQ veri seti çekişmeli üretici ağlar için hazırlanmış yaş ve farklı etnik köken çeşitliliğine sahip yüksek çözünürlüklü görüntülerden oluşan veri setidir [19]. StyleGAN2 veriye dayalı koşulsuz üretken modellemede oldukça başarılı sonuçlar vermektedir [20]. Bu nedenle StyleGAN2 ile oluşturulan sahte yüzler oldukça gerçekçi ve zorlayıcıdır. Veri setindeki görüntülerin %80'i eğitim ve geçerleme işlemi için kullanılırken %20'si test işlemi için kullanılmıştır [21].

7.3. Hiper Parametreler

Modelin eğitilmesi için Çizelge 7.1'de yer alan hiper parametreler kullanılmıştır;

Çizelge 7.1. Eğitimde kullanılan hiper parametreler

Hiper Parametre	Değer
devir (epoch)	15
devir için adım sayısı (steps_per_epoch)	64
ayrıntı (verbosity)	1
öğrenme oranı (learning rate)	0,001
kayıp fonksiyonu (loss_function)	binary_crossentropy
iyileştirici (optimizer)	Adam

Çizelge 7.1'de yer alan değerlerde önerilen model maksimum başarı oranına ulaşmıştır. Devir değeri veri setinin kaç kez önerilen evrişimsel sinir ağında eğitildiğinin göstermektedir. Yapılan çalışmada veri seti 15 defa önerilen evrişimsel sinir ağında eğitildiğinde maksimum değere ulaşılmıştır. Veri seti sayısı yüksek olduğu için devir için adım sayısı değeri 64 olarak ayarlanmış ve veri seti her bir adımda 64 parça halinde alınarak eğitilmiştir. Model eğitiminin hangi aşamada olduğunun görüntülenebilmesi için ayrıntı değeri 1 olarak ayarlanmıştır. Model eğitiminde öğrenilen ağırlıkların güncellenme oranı olan öğrenme oranı 0,001 olarak ayarlanmıştır. Modelin tahmini ile gerçek değer arasındaki kaybı hesaplayabilmek için ikili bir sınıflandırma yapıldığı için ikili çapraz entropi

fonksiyonu kullanılmıştır. Belirlenen öğrenme oranının farklı parametrelere göre güncellenebilmesi için Adam optimizasyon algoritması kullanılmıştır. Adam optimizasyon algoritması hesaplama açısından verimli olması, düşük bellek gereksinimine ihtiyaç duyması ve veri seti büyük problemler için uygun olması nedeniyle tercih edilmiştir [22].

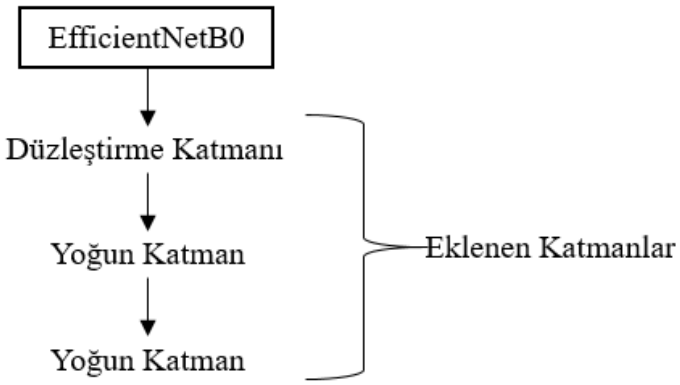
7.4. Deneyler ve Değerlendirmeler

Yapılan çalışmada Python 3.6 ve Tensorflow yazılım kütüphanesi kullanılmıştır. Eğitim ve test işlemi için 70.000 sahte ve 70.000 gerçek yüz görüntüsü olmak üzere toplam 140.000 görüntü içeren veri seti kullanılmıştır. Veri setinin %80'i eğitim %20'si test için kullanılmıştır. Transfer öğrenme yöntemi ile ImageNet üzerinde ön eğitilmiş modeller üzerine eğitim işlemleri gerçekleştirilmiştir. Modelin mobil ve gömülü cihazlarda maksimum performansta çalışabilmesini sağlamak için hafif evrişimsel sinir ağları tercih edilmiştir. Yapılan çalışmada mobil ve gömülü cihazlarda iyi sonuçlar vererek kendini kanıtlamış MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile evrişimsel sinir ağları kullanılmıştır. İlk aşamada bu algoritmalar transfer öğrenme ile kullanılırken diğer katmanlar dondurularak bir düzleştirme, ReLU aktivasyon fonksiyonuna sahip bir yoğun ve iki düğümden oluşan bir yoğun katman eklenmiştir. Modeller eğitildikten sonra EfficientNetB0 evrişimsel sinir ağı %93,64 doğruluk oranı ile en yüksek doğruluk oranına ulaşmıştır [51]. EfficientNetB0 algoritmasından transfer öğrenme için yapılan değişiklik Şekil 7.2'de görülmektedir.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b0 (Functional  (None, 1280)             4049564
 )

 flatten_5 (Flatten)         (None, 1280)              0

 dense_6 (Dense)             (None, 128)               163968

 dense_7 (Dense)             (None, 2)                 258

=================================================================
Total params: 4,213,790
Trainable params: 4,171,774
Non-trainable params: 42,016
_________________________________________________________________
```

Şekil 7.2. Transfer öğrenme için EfficientNetB0 ağında yapılan değişiklik

EfficientNetB0 algoritmasına eklenen katmanlar Şekil 7.3'de görülmektedir.



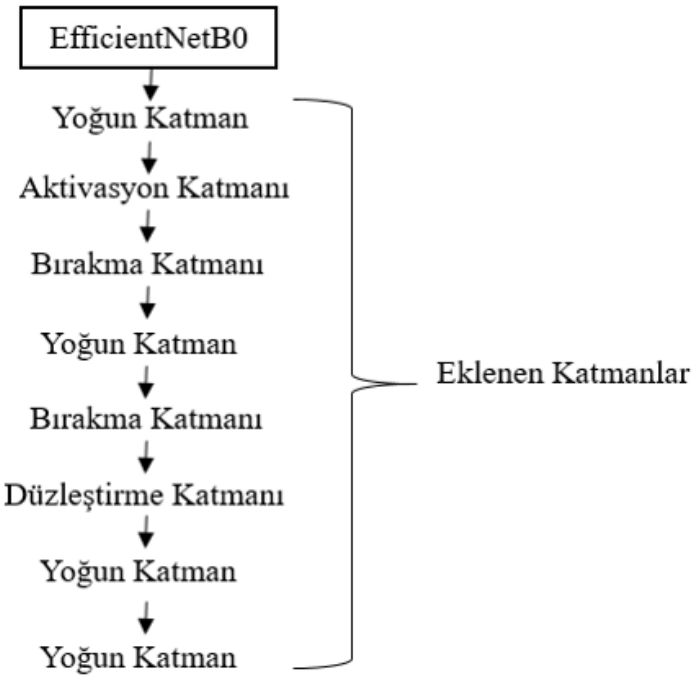
Şekil 7.3. EfficientNetB0 algoritmasına eklenen katmanlar

Başarı oranını artırabilmek için en yüksek başarı oranına ulaşılan EfficientNetB0 algoritmasında transfer öğrenme için eklenen katman sayısı parametre sayısı minimum seviyede değişecek şekilde artırılmıştır. EfficientNetB0 evrişimsel sinir ağı transfer öğrenme ile önceki katmanlar dondurulurken ReLU aktivasyon fonksiyonuna sahip iki yoğun katman (256 nöron), iki bırakma katmanı, bir düzleştirme katmanı, ReLU aktivasyon fonksiyonuna sahip bir yoğun katman (128) ve sınıflandırma için kullanılan softmax aktivasyon fonksiyonuna sahip iki düğümlü yoğun katman eklenmiştir. Bu işlem sonucu parametre sayısı 4,476,446 olmuştur. Yeni revize edilmiş model ile %95,48 doğruluk oranına ulaşılmıştır. EfficientNetB0 algoritmasından transfer öğrenme için yapılan yeni değişiklik Şekil 7.4'de görülmektedir.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b0 (Functional  (None, 1280)             4049564
 )

 dense_20 (Dense)            (None, 256)               327936

 activation_4 (Activation)   (None, 256)               0

 dropout_8 (Dropout)         (None, 256)               0

 dense_21 (Dense)            (None, 256)               65792

 dropout_9 (Dropout)         (None, 256)               0

 flatten_9 (Flatten)         (None, 256)               0

 dense_22 (Dense)            (None, 128)               32896

 dense_23 (Dense)            (None, 2)                 258

=================================================================
Total params: 4,476,446
Trainable params: 4,434,430
Non-trainable params: 42,016
_________________________________________________________________
```

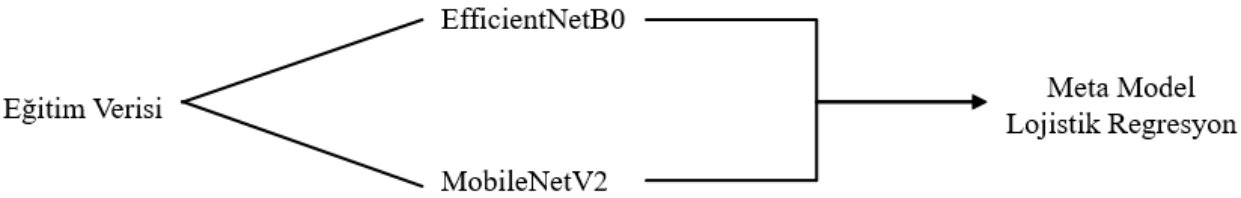
Şekil 7.4. EfficientNetB0 algoritmasından transfer öğrenme için yapılan yeni değişiklik

EfficientNetB0 algoritmasına eklenen yeni katmanlar Şekil 7.5'de görülmektedir.



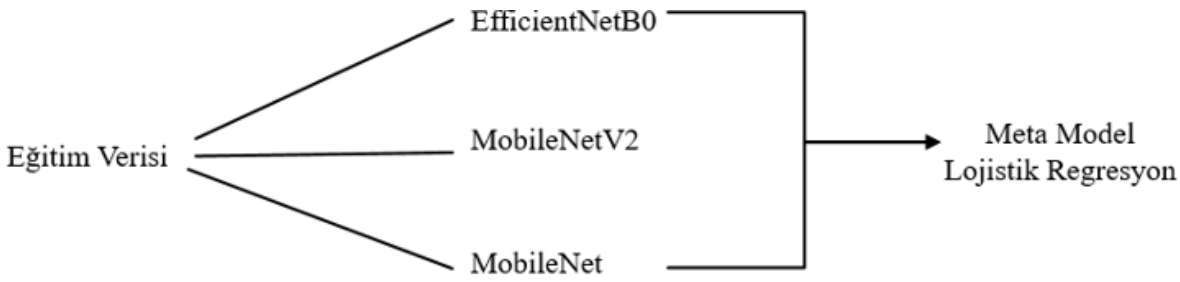
Şekil 7.5. EfficientNetB0 algoritmasına eklenen yeni katmanlar

Geliştirilmiş yeni EfficientNetB0 modeli ile elde edilen başarı oranını geliştirmek amacıyla son dönemlerde yoğun olarak kullanılmaya başlanan topluluk öğrenme yöntemi kullanılmıştır. Bu kapsamda yığınlama adı verilen topluluk öğrenme yöntemi ile ilk olarak en iyi sonuç alınan EfficientNetB0 evrişimsel sinir ağı ile diğer en yüksek başarı oranına ulaşılan MobileNetV2 modeli birlikte kullanılarak eğitildiğinde başarı oranı %96,20 olmuştur. EfficientNetB0 ile MobileNetV2 modelleri lojistik regresyon ile birlikte tekrar eğitilmiştir. Gerçekleştirilen işlem Şekil 7.6'da görülmektedir.



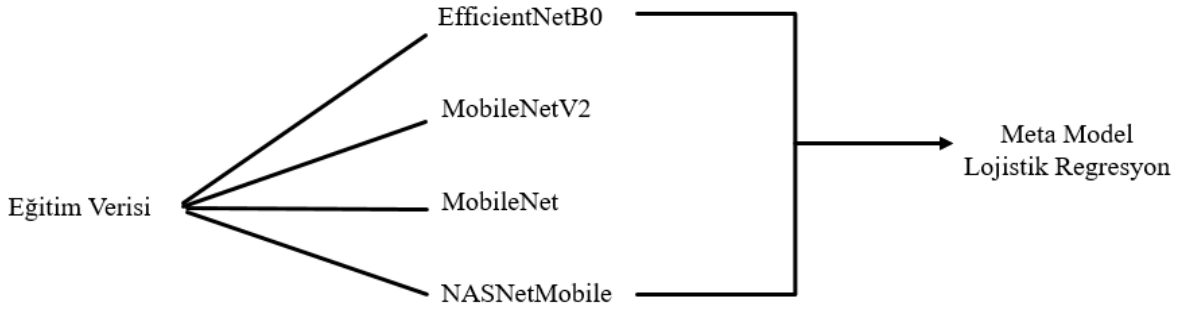
Şekil 7.6. EfficientNetB0 ile MobileNetV2 modellerinin birlikte kullanılması

Ardından MobileNet modeli de eklenerek üc model yığınlama topluluk öğrenme yöntemi ile kullanıldığında başarı oranı %96,41 olmuştur. EfficientNetB0, MobileNetV2 ve MobileNet modelleri lojistik regresyon ile birlikte tekrar eğitilmiştir. Gerçekleştirilen işlem Şekil 7.7'de görülmektedir.



Şekil 7.7. EfficientNetB0, MobileNetV2 ve MobileNet modellerinin birlikte kullanılması

Son olarak NASNetMobile modeli eklenerek dört model yığınlama topluluk öğrenme yöntemi ile kullanıldığında başarı oranı %96,27 olmuştur. EfficientNetB0, MobileNetV2 MobileNet ve NASNetMobile modelleri lojistik regresyon ile birlikte tekrar eğitilmiştir. Gerçekleştirilen işlem Şekil 7.8'de görülmektedir.



Şekil 7.8. EfficientNetB0, MobileNetV2, MobileNet ve NASNetMobile modellerinin birlikte kullanılması

Yapılan tüm denemelerin sonuçları Çizelge 7.2'de görülmektedir.

Çizelge 7.2. Sahte yüz görüntüsü tespiti modellerinin performans metrikleri

Algoritma	Doğruluk (Accuracy) (%)	Kesinlik (Precision) (%)	Duyarlılık (Recall) (%)	F1 SKOR (%)	Parametre Sayısı
Geliştirilmiş EfficientNetB0 + MobileNetV2+MobileNet	96,44	97,82	97,36	97,58	-
Geliştirilmiş EfficientNetB0 + MobileNetV2+MobileNet+ NASNetMobile	96,27	96,25	96,27	96,25	-
Geliştirilmiş EfficientNetB0 + MobileNetV2	96,20	97,59	96,07	96,83	-
Geliştirilmiş EfficientNetB0	95,48	96,14	95,43	95,78	4,476,446
EfficientNetB0	93,64	93,70	93,27	93,48	4,213,790
MobileNetV2	91,12	92,53	91,87	92,19	3,500,000
MobileNet	87,83	93,66	87,25	90,34	4,253,864
NASNetMobile	78,87	79,12	78,04	78,57	5,600,000

Çizelge 7.2'de görüldüğü gibi geliştirilmiş EfficientNetB0, MobileNet ve MobileNetV2 modelleri yığınlama topluluk öğrenme modeli ile tekrar kullanıldığında %96,44 ile en yüksek doğruluk oranına ulaşmıştır. Önerilen modelin önceki çalışmalarla karşılaştırılması Çizelge 7.3'de görülmektedir.

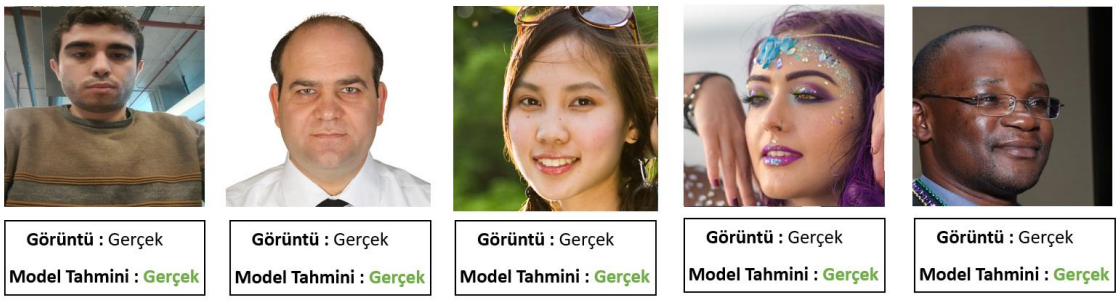
Çizelge 7.3. Önerilen modelin önceki çalışmalarla karşılaştırılması

Çalışma	Teknik	Algoritma	Özellik	Veri Seti	Değerlendirme Metriği	Başarı Oranı
Önerilen Model	Derin öğrenme	Geliştirilmiş EfficientNetB0 + MobileNetV2 +MobileNet	Sahte yüz tespiti	FFHQ ve StyleGAN2	Doğruluk	%96,44
Wang vd. [2]	Derin öğrenme	Nöron izlemesine dayalı yeni model	Sahte yüz tespiti	FFHQ ve StyleGAN2	Doğruluk	%91,9
St vd. [9]	Derin öğrenme	FisherFace	Sahte yüz tespiti	FFHQ	Doğruluk	%94,92
Bang vd. [10]	Derin öğrenme	MobileNetV3	Sahte yüz tespiti	FFHQ ve StyleGAN2	Doğruluk	%83,64
Hu vd. [11]	Görüntü işleme	Dlib kütüphanesi	Sahte yüz tespiti	FFHQ ve StyleGAN2	AUC	%94
Guo vd. [12]	Derin Öğrenme	EfficientNet-B5	Sahte yüz tespiti	FFHQ ve StyleGAN2	AUC	%91

Çizelge 7.3'de görüldüğü gibi önerilen model tansfer öğrenme ve topluluk öğrenme yöntemlerini kullanması sayesinde önceki çalışmalardan daha yüksek doğruluk oranına ulaşmıştır. Modelin sahte görüntüler üzerinde yaptığı başarılı tahminler Şekil 7.9'da görülmektedir.

Şekil 7.9. Önerilen modelin sahte yüz görüntüleri üzerinde yaptığı tahminler

Şekil 7.9'da görüldüğü gibi önerilen model olukça gerçekçi üretilen ve çeşitli insan yüzleri üzerinde doğru tahmin yapabilmektedir. Modelin gerçek görüntüler üzerinde yaptığı başarılı tahminler Şekil 7.10'da görülmektedir.



Şekil 7.10. Önerilen modelin gerçek yüz görüntüleri üzerinde yaptığı tahminler

Şekil 7.10'da görüldüğü gibi önerilen model çeşitli ve zorlayıcı gerçek insan yüzleri üzerinde doğru tahmin yapabilmektedir. Modelin test edilen çeşitli görüntüler üzerinde yaptığı yanlış tahminler Şekil 7.11'de görülmektedir.

Şekil 7.11. Önerilen modelin yüz görüntüleri üzerinde yaptığı yanlış tahminler

Şekil 7.11'de görüldüğü gibi model bazı yüz görüntüleri üzerinde yanlış tahminler yapmıştır. Model görüntü 1'de filtre bulunması nedeniyle görüntüyü sahte olarak sınıflandırmıştır. Yüz üzerinde doğal olmayan şekiller çizildiğinden görüntü 2 model tarafından yanlış sınıflandırılmıştır. Veri setinde çocuk yüz görüntü sayısının daha az olması nedeniyle model görüntü 3'ü yanlış sınıflandırmıştır. Yüz görüntüsü üzerinde makyajla veya aksesuarla fazla

değişiklik yapılması nedeniyle model görüntü 3'ü yanlış sınıflandırmıştır. Görüntü 5'teki yüz görüntüsünde yer alan güneş gözlüğü modelin yüzün tamamına göre analiz edilmesini engellediği için modelin yanlış tahmin yapmasına neden olmuştur.

8. SONUÇ VE ÖNERİLER

Sahte yüz görüntüleri makine öğrenmesi, görüntü işleme vb. teknikler kullanılarak üretilen içeriklerdir. Günümüzde en yaygın dijital manipülasyon türüdür. Sahte yüz görüntülerinin çoğu kişileri itibarsızlaştırma veya dolandırıcılık amaçlıdır. Sahte yüz görüntüleri ile kişiler bulunmadıkları yerlerde gösterilebilir veya kişilere söylemedikleri sözlerle konuşma yaptırılabilir. Sahte yüz üretiminde makine öğrenmesi algoritmalarının kullanılması içeriklerin gerçek görüntülerden ayırt edilmesini giderek zorlaştırmaktadır. Yapılan çalışmada sahte yüz görüntülerinin tespiti için MobileNet, MobileNetV2, EfficientNetB0 ve NASNetMobile hafif evrişimsel sinir ağları kullanılmıştır. Mobil cihazların yaygın kullanılması nedeniyle yapılan çalışmada mobil cihazlarda çalışabilen hafif evrişimsel sinir ağları tercih edilmiştir. Eğitim işlemi için 70.000 gerçek ve 70.000 sahte görüntü içeren veri seti kullanılmıştır. EfficientNetB0 algoritması yapılan eğitim işlemleri sonucunda en yüksek doğruluk oranına ulaşmıştır. Doğruluk oranını artırabilmek için EfficientNetB0, MobileNet ve MobileNetV2 topluluk öğrenme yöntemi ile birlikte kullanıldığında %96,44 ile en yüksek doğruluk oranına ulaşılmıştır. Geliştirilen model ile kullanıcılar mobil cihazları üzerinde görüntülerdeki yüzlerin gerçek mi yoksa sahte mi olduğunu hızlı bir şekilde kontrol edebilir. Bu sayede sahte içeriklerin ve yanlış bilgilerin yayılmasının engellenmesine katkı sağlanır.

Gerçek görüntülere filtreler uygulanması, yüzlerde doğal olmayan şekiller bulunması, ağır makyaj, yüzde gözlük, metal vb. aksesuar bulunması ve çocuk görüntüleri modelin yanlış çıkarım yapmasına neden olmaktadır. Gelecek çalışmalarda modelin yanlış tahminler yaptığı durumların engellenmesi için veri setinin çeşitliliğinin artırılması model başarısının önemli ölçüde artırılmasını sağlayacaktır. Bunun yanında sonraki çalışmalarda modele kimlik değiştirme, nitelik manipülasyonu ve ifade değiştirme manipülasyon tespiti transfer öğrenme ile eklenerek 4 temel yüz manipülasyonunun tespit edilmesi sağlanabilir.

KAYNAKLAR

1. Pashine, S., Mandiya, S., Gupta, P. ve Sheikh, R. (2021). Deep fake detection: survey of facial manipulation detection solutions. *International Research Journal of Engineering and Technology*, 8(5), 4441-4449.

2. Wang, R., Juefei-Xu, F., Ma, L., Xie, X., Huang, Y., Wang, J. ve Liu, Y. (2020). *FakeSpotter: a simple yet robust baseline for spotting ai-synthesized fake faces.* IJCAI'20: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Sanal, 3444-3451.

3. Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A. ve Ortega-Garcia (2020). Deepfakes and beyond: s survey of face manipulation and fake detection. *Information Fusion*, 131-148.

4. Choi, Y., Choi, M., Kim, M., Ha. J.-W., Kim, S. ve Choo, J. (2018). *StarGAN: unified generative adversarial networks for multi-domain image-to-image translation.* IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 8789-8797.

5. Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C. ve Nießner, M. (2016). Face2face: real-time face capture and reenactment of rgbvideos. *Communications of the ACM*, 62(1), 96-104.

6. Thies, J., Zollhofer, M. ve Nießner, M. (2019). Deferred neural rendering: image synthesis using neural textures. *ACM Transactions on Graphics*, 38(4), 1-12.

7. Öcal, H., Doğru, İ.A. ve Barışçı, N. (2019). Akıllı ve geleneksel giyilebilir sağlık cihazlarında nesnelerin interneti. *Politeknik Dergisi*, 22(3), 695-714.

8. Şafak, E., Arslan, Ç., Gözütok, M. ve Köprülü, T. (2021). Dağıtık defter teknolojileri ve uygulama alanları üzerine bir inceleme. *Avrupa Bilim ve Teknoloji Dergisi*, 36-45.

9. St, S., Ayoobkhan, M.U.A., Kumar, K., Bacanin, N., Venkatachalam, K., Štěpán, H. ve Pavel, T. (2022). Deep learning model for deep fake face recognition and detection. *PeerJ Computer Science*, 8(2).

10. Bang, Y.O. ve Woo, S.S. (2021). DA-FDFtNet: dual attention fake detection fine-tuning network to detect various ai-generated fake images. *CoRR*, abs/2112.12001.

11. Hu, S., Li, Y. ve Lyu, S. (2021). *Exposing GAN-generated faces using inconsistent corneal specular highlights.* International Conference on Acoustics, Speech and Signal Processing, Toronto, 1-6.

12. Guo, H., Hu, S., Wang, X., Chang, M.C. ve Lyu, S. (2022). *Eyes tell all: irregular pupil shapes reveal gan-generated faces.* IEEE International Conference on Acoustics, Speech and Signal Processing, Singapore, 2904-2908.

13. Tasova, O. (2017). *Yapay sinir ağları ile yüz tanıma*. Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, İzmir.

14. Haykin, S. (1999). *Neural networks: a comprehensive fundation.* (2. Baskı). Kanada: Pearson Prentice Hall, 96.

15. Kaya, E. (2022). İleri beslemeli yapay sinir ağının eğitiminde meta-sezgisel yaklaşımlar. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, 15(1), 38-43.

16. Şafak, E. (2018). *Evrişimsel sinir ağlarını kullanarak yaş ve cinsiyet tahmini.* Gazi Üniversitesi Teknoloji Fakültesi Lisans Tezi, Ankara.

17. Albawi, S., Mohammed, T. A. ve Al-Zawi S. (2017). *Understanding of a convolutional neural network.* International Conference on Engineering and Technology, Antalya, 1-6.

18. Şafak, E., Doğru, İ. A., Barışçı, N. ve Toklu, S. (2022). Derin öğrenme kullanılarak nesnelerin interneti tabanlı mobil sürücü yorgunluk tespiti. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 37(4), 1869-1882.

19. Şeker, A., Diri, B. ve Balık, H.H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. *Gazi Mühendislik Bilimleri Dergisi*, 3(3), 47-64.

20. İnik, Ö.,Ülker, E. (2017). Derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, 6(3), 85-104.

21. Khan, A., Sohail, A., Zahoora, U. ve Qureshi, A.S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 5455–5516.

22. Lecun, Y., Bottou, L., Bengio, Y. ve Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278 - 2324.

23. Krizhevsky, A., Sutskever, I. ve Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(2).

24. Simonyan, K. ve Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

25. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. ve Rabinovich, A. (2015). *Going deeper with convolutions.* IEEE Conference on Computer Vision and Pattern Recognition, Boston, 1-9.

26. He, K., Zhang, X., Ren, S. ve Sun, J. (2016). *Deep residual learning for image recognition.* IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 770-778.

27. Chollet, F. (2017). Xception: deep learning with depthwise separable convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. ve Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861.

28. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. ve Adam, H. (2017). MobileNets: efficient convolutional neural networks for mobile vision applications. *CoRR,* abs/1704.04861.

29. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. ve Chen, L.C. (2018). *MobileNetV2: inverted residuals and linear bottlenecks.* IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, 4510-4520.

30. Tan, M. ve Le, Q.V. (2019). *EfficientNet: rethinking model scaling for convolutional neural networks.* 36th International Conference on Machine Learning, California, 1-11.

31. Saxen, F., Werner, P., Handrich, S., Othman, E., Dinges, L. ve Al-Hamadi, A. (2019). *Face attribute detection with mobilenetv2 and nasnet-mobile.* 11th International Symposium on Image and Signal Processing and Analysis, Dubrovnik, 176-180.

32. İnternet: Tensorflow. URL: https://www.tensorflow.org, Son Erişim Tarihi: 07.01. 2023.

33. Janardhanan, P. (2020). Project repositories for machine learning with tensorFlow. *Procedia Computer Science*, 188-196.

34. Şeker, A., Diri, B. ve Balık, H.H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. *Gazi Mühendislik Bilimleri Dergisi*, 3(3), 47-64.

35. Al-Rfou, R., Alain, G., Almahairi, A. vd. (2016). Theano: a python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688.

36. Harris, C.R., Millman, K.J., van der Walt, S.J. vd. (2020). Array programming with numpy. *Nature*, 357–362.

37. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. ve Duchesnay, É. (2012). Scikit-learn: machine learning in python. *The Journal of Machine Learning Research*, 2825-2830.

38. İnternet: DeepLearning4j. URL: https://deeplearning4j.konduit.ai, Son Erişim Tarihi: 07.01.2023.

39. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. ve Zhang, Z. (2015). MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274.

40. İnternet: Keras. URL: https://keras.io, Son Erişim Tarihi: 07.01.2023.

41. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. ve Chintala, S. (2019). *PyTorch: an imperative style, high-performance deep learning library*. NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, 1-12.

42. Rincy, T.N. ve Gupta, R. (2020). *Ensemble learning techniques and its efficiency in machine learning: a survey.* 2nd International Conference on Data, Engineering and Applications, Bhopal, 1-6.

43. Mienye, I.D. ve Sun Y. (2022). A survey of ensemble learning: concepts, algorithms, applications, and prospects. *IEEE Access*, 99129-99149.

44. Ganaie, M.A., Hu, M., Malik, A.K., Tanveer, M. ve Suganthan, P.N. (2022). Ensemble deep learning: a review. *Engineering Applications of Artificial Intelligence*, 115.

45. Wen, L. ve Hughes, M. (2020). Coastal wetland mapping using ensemble learning algorithms: a comparative study of bagging, boosting and stacking techniques. *Remote Sensing*, 12(10).

46. Bozinovski, S. ve Fulgosi, A. (1976). *The influence of pattern similarity and transfer of learning upon training of a base perceptron b2.* Processing Symposium Informatica 3-121-5, Bled, 1-12.

47. Ribani, R. ve Marengoni, M. (2019). *A survey of transfer learning for convolutional neural networks.* 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials, Rio de Janeiro, 47-57.

48. Agarwal, N., Sondhi, A., Chopra, K. ve Singh, G. (2020). Transfer learning: survey and classification. *Smart Innovations in Communication and Computational Sciences*, 1168, 145-155.

49. Weiss, K., Khoshgoftaar, T.M. ve Wang D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1).

50. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. ve He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43-76.

51. Şafak E. ve Barışçı N. (2022). Hafif evrişimsel sinir ağları kullanılarak sahte yüz görüntülerinin tespiti. *El-Cezeri*, 9(4), 1282-1289.

Gazili olmak ayrıcalıktır