İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

CNN Derin Öğrenme Algoritmasıyla Yüz İfadelerinin Değerlendirilmesi

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Ömer KAPÇAK

Y220234045

Proje Danışmanı: Dr.Öğr. Üyesi Serpil YILMAZ

Ocak 2024

CNN Derin Öğrenme Algoritmasıyla Yüz İfadelerinin Değerlendirilmesi

Öz

Yüz ifadelerinin incelenmesi geçmişten bugüne dek birçok farklı alanda inceleme konusu olmuştur. Günümüz teknolojisinde yüz ifadelerinin oluşturduğu duyguların tespiti özellikle psikoloji alanı olmak üzere robotik, sağlık, bilgisayar oyunları, trafik, sanal gerçeklik vb. alanlarda eğlence, konfor, güvenlik ve hayatı kolaylaştırma gibi birçok nedenle kullanılmaktadır. Bir kişinin duygusal tepkilerinin fiziksel özelliklerine yansımasından birçok çıkarım yapılabilmektedir.

Bu çalışmada Kaggle.com web sitesinde bulunan, 35.685 adet 48x48 piksel tek renk yüz ifadelerini içeren "Emotion Detection" veri setinin CNN derin öğrenme algoritması yardımıyla çeşitli yüz ifadelerini (mutluluk, nötr, üzüntü, öfke, şaşkınlık, tiksinti, korku) sınıflandıran bir modelin oluşturulması amaçlanmıştır. Bu sınıflandırma yapılırken Confusion matrix, accuracy score, ROC etc. değerlendirme ölçütleri yardımıyla oluşturulan modelin değerlendirilmesi ve yüz ifadesi içeren herhangi bir fotoğrafın tahmininin yapılması hedeflenmektedir.

Anahtar Sözcükler: Yüz ifadelerinin sınıflandırılması, Yapay zekâ, Yapay sinir ağları, CNN derin öğrenme algoritması

Evaluation of Facial Expressions with CNN Deep Learning

Abstract

Examination of facial expressions has been the subject of investigation in many different fields from past to present. In today's technology, the detection of emotions created by facial expressions has many applications, especially in the field of psychology, robotics, health, computer games, traffic, virtual reality, etc. It is used in areas for many reasons such as entertainment, comfort, security and making life easier. Many inferences can be made from the reflection of a person's emotional reactions on their physical characteristics.

In this study, the "Emotion Detection" dataset, which contains 35,685, 48x48 pixel single color facial expressions on the Kaggle.com website, was used to develop a model that classifies various facial expressions (happiness, neutral, sadness, anger, surprise, disgust, fear) with the help of the CNN deep learning algorithm is intended to be created. Confusion matrix, accuracy score, ROC etc. are used while making this classification. The aim is to evaluate the model created with the help of evaluation criteria and to predict any photograph containing facial expressions.

Keywords: Classification of Facial Expressions, Artificial İntelligence, Artificial Neural Networks, CNN Deep Learning Algorithm

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

CNN	Convolution Neural Network
ReLU	Rectified Linear Unit
AI	Artificial Intelligence
SVM	Destek Vektör Makinesi
KNN	K-En Yakın Komşu Algoritması
DNN	Deep Neural Networks
LSTM	Uzun-Kısa Süreli Bellek
IoT	Nesnelerin İnterneti
API	Uygulama Programlama Arayüzü
GPU	Grafik İşlem Birimi

Bölüm 1

Giriş

Yüz ifadelerinin incelenmesi geçmişten bugüne dek birçok farklı alanda inceleme konusu olmuştur. Günümüz teknolojisinde yüz ifadelerinin oluşturduğu duyguların tespiti özellikle psikoloji alanı olmak üzere robotik, sağlık, bilgisayar oyunları, trafik, sanal gerçeklik vb. alanlarda eğlence, konfor, güvenlik ve hayatı kolaylaştırma gibi birçok nedenle kullanılmaktadır. Bir kişinin duygusal tepkilerinin fiziksel özelliklerine yansımasından birçok çıkarım yapılabilmektedir.

Psikoloji alanında birçok çalışmada duygular temel ve karmaşık olmak üzere iki sınıfta incelenir. Her ne kadar iki adet duygu sınıfı bulunmakta olsada bu konu üzerindeki yapay zekâ araştırmaları temel duygular üzerine yoğunlaşmıştır. Temel duygular mutlu (happy), üzgün (sad), öfkeli (angry), şaşırmış (surprised), iğrenmiş (disgusted), doğal (neutral) ve korkulu (fearful) yüz ifadeleridir.

Derin öğrenme algoritma çalışmalarından önce birçok geleneksel yöntemlerle ön işlem ve özellik çıkarım ile kullanılan makine öğrenmesi yöntemi ile çalışmalar gerçekleştirilmiştir. Fakat herhanfi bir sonuç elde edilememiştir.

Yüz ifadesi çalışmalarında, SVM yöntemi verdiği başarılı sonuçlar nedeni ile sınıflandırma amaçlı sıklıkla kullanılmaktadır. Liao ve ekibi farklı duyguların farklı yüz bölgelerindeki ağırlıklarını hesaplayarak yüz ifadesi tespiti yapmıştır Çalışmalarında nötr fotoğraf ile duygu ifadesi olan fotoğraf arasındaki optik akış incelenmiş olup, sınıflandırma için SVM kullanılmış ve ortalama %92,5 başarım elde edilmiştir [1].

Literatürde, KNN makine öğrenmesi algoritması duygu tanımada yaygın olarak kullanılmaktadır KNN makine öğrenmesi kullanılan bir çalışmada doğruluk%85'ten

fazla elde edilmiştir. Fakat KNN'nin uygulanması yüksek bellek gerektirir ve performans açısından ise oldukça yavaştır. Öte yandan, derin öğrenme temelli teknik, evrişimsel sinir ağı (CNN) yüksek doğruluk performansı ve hız sunmaktadır.

Tıptaki bir çalışmada ise Tripathi, EEG tabanlı duygu tespiti için Deep Neural Networks (DNN) ve CNN yöntemlerini önermiş değerlik ve uyarılma(dürtü) koşullarına göre sırasıyla %75,58 ve %73,28 sınıflandırma başarısı elde etmiştir. Bu deneysel çalışmalar, özellikle hız bakımından derin öğrenme modellerinin geleneksel yöntemlerden daha iyi performans gösterdiğini göstermiştir [2].

Daha sonrasında derin öğrenme algoritlarının kullanımı, 2015 yılında Chen ve arkadaşları, görüntü verisinden duygu analizi için oluşturdukları Evrişimli Sinir Ağı (CNN) modeli ile yaygınlaşmıştır [3].

Sepas-Moghaddam vd. ışık alanı kameralarından elde edilen iki boyutlu görüntüleri VGG-16 ESA ve uzun-kısa süreli bellek (LSTM) tabanlı ağ kullanarak hem uzamsal hem de açısal bilgileri modelleyebilen bir derin öğrenme yaklaşımı önermişlerdir. Önerilen model sayesinde 4 farklı duyguyu ortalama %75'in üzerinde bir başarı ile ayırt edebilmişlerdir [4].

Taghi Zadeh vd. insan duygularının tanınması için derin öğrenmeye dayalı bir yöntem önermişlerdir. Geliştirdikleri yöntem özellik çıkarma için Gabor filtrelerini ve ardından sınıflandırma için bir ESA mimarisini kullanmaktadır. ESA girdileri olarak internet erişimli bir veri setinden elde edilen yüz görüntülerini kullanmışlar ve altı farklı yüz duygu durumunu ayırmada %97 başarı elde etmişlerdir [5].

Bölüm 2

Yapay Zekâ

Son zamanlarda yapay zekâ kavramı oldukça popülerleşmeye başlamıştır. Bunun en önemli nedenleri arasında şüphesiz yapay zekâ ile beraber yaşamı kolaylaştıracak keşiflerin insanoğlunu oldukça heyecanlandırmasıdır. Bu heyecanın temeline indiğimizde ise karşımıza insanoğlunun geçmişten günümüze kadar süregelen cansız ve herhangi bir bilince sahip olmayan varlıkları bir şekilde harekete geçirme isteği çıkacaktır.

Yapay zekâ çalışmalarının uygulama alanı bulduğu en temel alanların başında insansı robotlar gelmektedir. Yapılan keşiflerle ve araştırmalarla teknolojinin sınırı genişledikçe insanoğlunun yerini doldurmayı amaçlayan insansı robotların ortaya çıkışı da zamanla artmaktadır.

Yapay zekâ alanı çok disiplinli alan olup, bilgisayar mühendisliği, programlama, elektronik, mekatronik ve felsefe gibi farklı alanların çalışma alanlarına dâhil olmaktadır.

2.1 Yapay Zekâ Nedir?

Yapay zekâ, bilgisayar destekli bir cihazın insana özgü niteliklerini, analizlerini, çözüme ulaşma şeklini, bir anlam çıkarma yeteneğini ve geçmişteki deneyimlerinden öğrenme gibi görevleri yerine getirme yeteneği olarak tanımlanmıştır [6].

Yapay zekâ için bilim dünyasında farklı tanımlamalarda kendine yer bulmuştur. Slage'ye göre yapay zekâ, sezgisel programlama temelinde olan bir yaklaşımdır [7]. Popov'a göre yapay zekâ, insanların yaptıklarını bilgisayarlara yaptırabilme çalışmasından ibarettir [8].

Yapay zekâ mantığının temelinde öğrenebilme yeteneği yatar. Nasıl bir insan bir işi yapabilmek için önce o işi nasıl yapılacağını öğrenmesi gerekiyorsa yapay zekânın da yapacağı işi önce öğrenmesi gerekir. Bu bağlamda yapay zekânın en büyük

faydalarından biri öğrendikleri en doğru yolu çok hızlı bir şekilde katetmesi olacaktır. Örneğin insan çabuk etkilenir, ancak uzman yapay zekâ kalıcıdır. Duygularıyla değil realisttik ve teknik olarak hareket eder.

2.2 Yapay Zekâ Teknolojileri

Yapay zekâ teknolojileri uzman sistemler, bulanık mantık, yapay sinir ağları, makine öğrenmesi ve genetik algoritmalarından oluşur.

2.2.1 Uzman Sistemler

Bir problemi o problemin uzmanların çözdüğü mantıkta hareket ederek çözebilme yeteneğine sahip bilgisayar programları geliştiren teknolojilerdir. Uzmanların bir probleme yaklaşma mantığını kullanırlar.

2.2.2 Makine Öğrenmesi

Bilgisayarların öğrenmesini sağlayan teknolojidir. Genellikle veri seti kullanarak girdi ve çıktılar arasındaki ilişkileri kullanarak bir öğrenme gerçekleştirirler.

2.2.3 Genetik Algoritmalar

Daha çok karmaşık optimizasyon problemlerinin çözülmesi için yararlanılan teknolojilerdir. Bir problemi çözmek için her şeyin başında rastgele başlangıç çözümleri belirlenmektedir. Daha sonra bu çözümler birbirleri ile eşleştirilerek performansı yüksek çözümler üretebilmektedir. Bir genetik algoritmasının temel elemanları Kromozom ve gen, çözüm, çaprazlama, mutasyon, uygunluk fonksiyonu ve yeniden üretimidir [9].

2.2.4 Bulanık Mantık

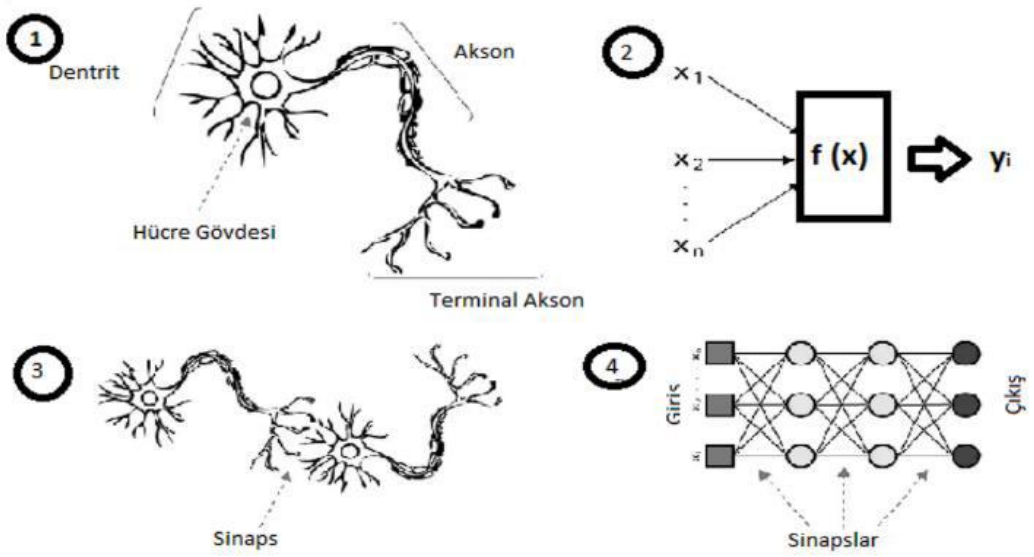
İnsanlar tarafından üretilmiş, günlük dilde kullanılan herhangi bir kesinlik bildirmeyen cümlelerin derecelendirilmesini sağlar. Örneğin, “biraz soğuk”, “hemen hemen doğru”, “çok yavaş” gibi belirsiz cümleler, problemlerin çözülmesine yardımcı

olmasına rağmen sayısal olarak ifade edilemezler. Bulanık mantık algoritması bu gibi durumlarda insan bilincini taklit ederek belirsiz çözümlemeler ve sayısal modellemeler ortaya çıkarır.

2.3 Yapay Sinir Ağları

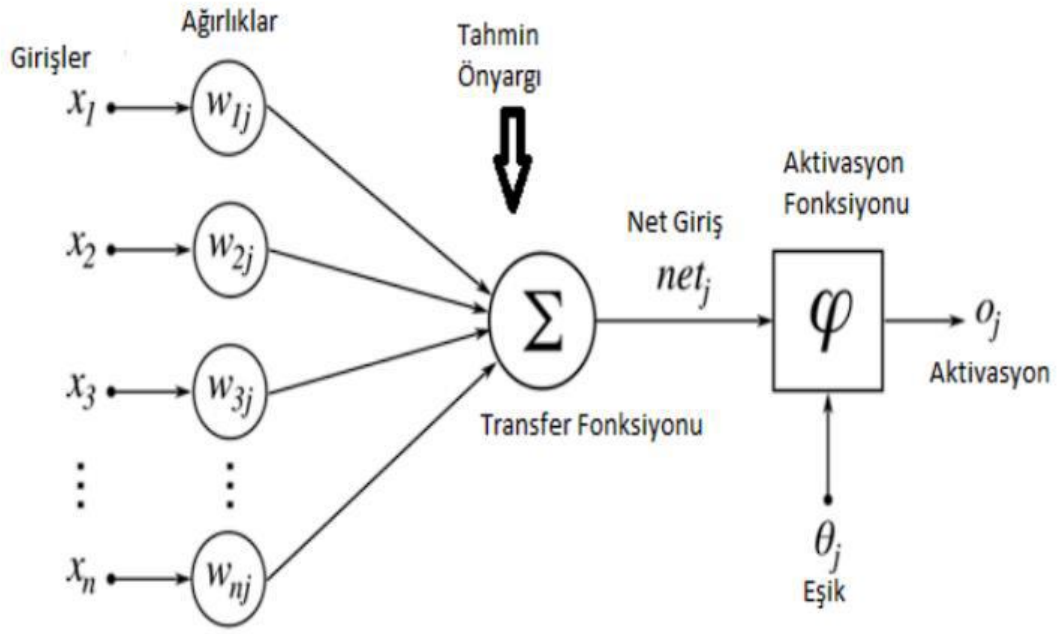
Yapay sinir ağları, insan beyninin öğrenme adımlarını taklit ederek zihnin öğrenme, hatırlama, genelleme yapma yolu ile topladığı verilerden yeni veri üretebilme gibi temel işlevlerin uygulanabildiği bilgisayar yazılımlarıdır. Yapay sinir ağları biyolojik sinir ağlarını taklit eden sentetik yapılardır.

2.3.1 Yapay Sinir Hücresi



Şekil 2.1: Yapay Sinir Ağı Modelleri

Aşağıdaki şekilde gözlemlenebileceği gibi bir hücreye n tane veri girişi yapılmaktadır. Girilen veriler ağırlıklar ile çarpılarak toplanır ve sonra ön yargı eklenir en nihayetinde ise net bir sonuca ulaşılır. Net girdi aktivasyon fonksiyonundan geçer ve sonucunda bir veri çıktısı elde edilmiş olur.



Şekil 2.2: Birden Çok Girdisi Olan Yapay Sinir Ağı Modeli

Yapay sinir ağlarının içinde yer alan tüm sinir hücreleri bir ya da birden fazla veri yani girdi alırken tek bir çıktı verirler. Bu çıktı tüm sinir ağının dışına verilen bir çıktı olabileceği gibi bir başka sinir hücresine aktarılan girdide olabilir. Bir yapay sinir hücresi beş temel bileşenden oluşmaktadır:

• Girdiler

• Ağırlıklar

• Toplama Fonksiyonu

• Aktivasyon Fonksiyonu

• Çıktı

2.3.1.1 Girdiler

Bir yapay sinir hücresine dışarıdan gelen bilgi olarak adlandırılır. Bu bilgiler ağın öğrenmesi istenen örnekler tarafından belirlenir. Daha önce bahsedildiği gibi bir yapay sinir hücresine dışarıdan bilgi girdi olarak gelebildiği gibi bir sinir hücresinden de aktarılabilir.

2.3.1.2 Ağırlıklar

Ağırlıklar, yapay sinir hücresi tarafından alınan girdileri sinir üzerindeki etkisini belirleyen ideal katsayılardır.

2.3.1.3 Toplama Fonksiyonu

Bu fonksiyon, hücreye gelen net girdiyi hesaplar. Toplama fonksiyonu için çeşitli fonksiyonlar kullanılmaktadır. En yaygın olanı ise ağırlıklı toplam bulmaktır. Bu aşamada gelen her girdi kendi ağırlığı ile çarpılarak toplanır. Böylece ağa gelen net girdi tespit edilmiş olunur.

$$NET = \sum_{i=1}^{n} G_i A_i$$

Şekil 2.3: Toplama Fonksiyonu

G Girdileri, A ağırlıkları, n ise hücreye gelen toplam girdi sayısını temsil etmektedir. Yapay sinir ağlarında kullanılan diğer toplama fonskiyonları:

• Çarpım Net Girdi

• Maksimum Net Girdi

• Minimum Net Girdi

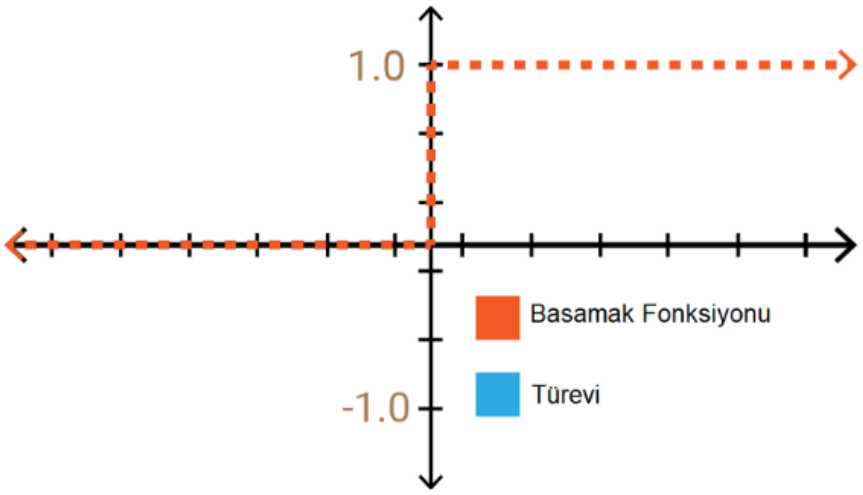
• Çoğunluk Net Girdi

• Kümülatif Toplam Net Girdi

2.3.1.4 Aktivasyon Fonksiyonu

Aktivasyon fonksiyonu, toplama fonksiyonundan gelen girdiyi işleyerek yapay sinir hücresinin üreteceği çıktıyı belirler. Yapay sinir ağlarına doğrusal olmayan gerçek dünya problemlerini doğru aktarabilmek için aktivasyon fonksiyonuna ihtiyaç duyarız. Eğer bir yapay sinir ağına aktivasyon fonksiyonu uygulanmazsa çıkış sinyali basit bir doğrusal fonksiyon haline gelir. Yani aktivasyon fonksiyonu kullanılmayan bir sinir ağı sınırlı öğrenme gücüne sahip bir lineer regresyon gibi davranacaktır. Ama yapay sinir ağlarında hedef doğrusal olmayan karmaşık durumlarında üstesinden gelmesini hedefliyoruz.

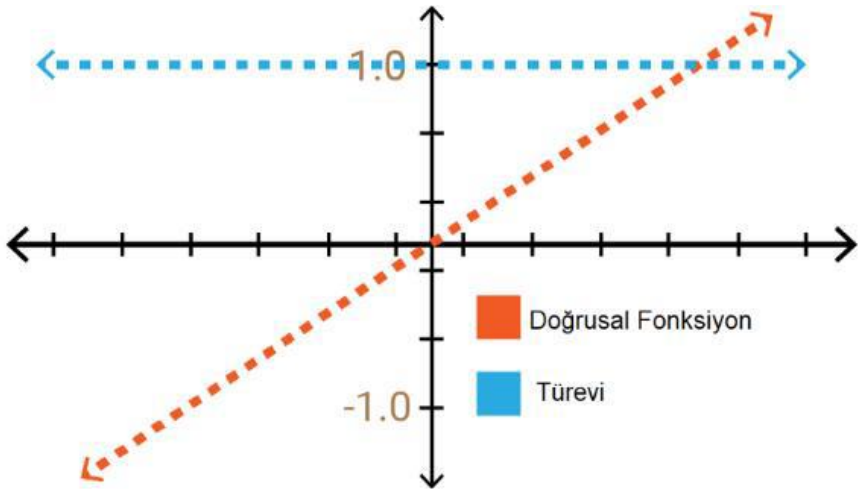
Basamak (Step) Fonksiyonu



Şekil 2.4: Basamak (Step) Fonksiyonu

İkili değer alan fonksiyondur. Tabiatı gereği ikili sınıflandırıcı olarak kullanılır. Bu yüzden genellikle çıkış katmanlarında tercih edilir. Türevlenemediği için gizli katmanlarda kullanılamaz.

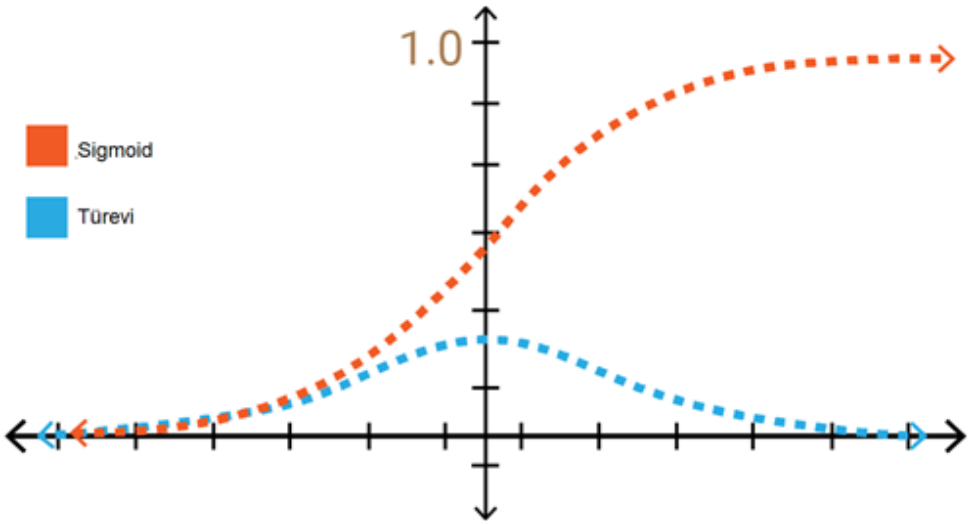
Doğrusal (Linear) Fonskiyon



Şekil 2.5: Doğrusal (Linear) Fonksiyon

Bir dizi aktivasyon değeri üretir ve bunlar basamak fonksiyonundaki gibi iki değerler değildir. Birkaç nöronu birbirine bağlamaya izin verir. Ancak bu aktivasyon fonksiyonunun sorunu türevinin sabit olmasıdır. Türevi sabit olduğu için doğru öğrenme işlemini gerçekleştiremeyebilir.

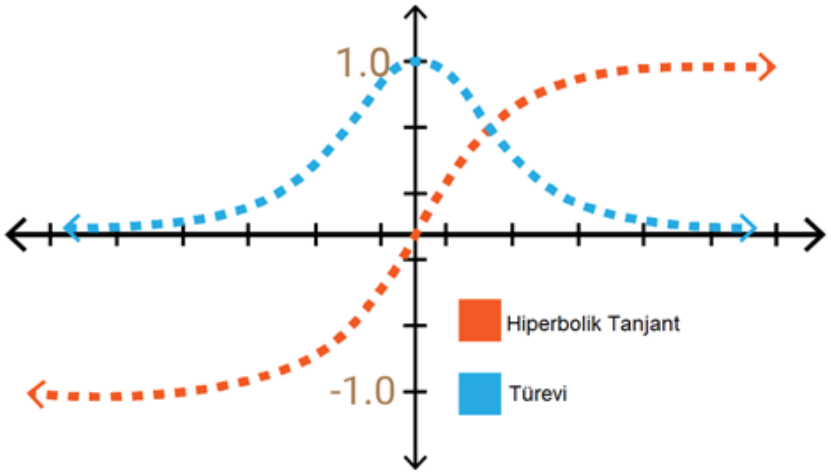
Sigmoid Fonksiyonu



Şekil 2.6: Sigmoid Fonksiyonu

Sigmoid fonksiyonu en sık kullanılan aktivasyon fonksiyonudur. Sigmoid fonksiyonun kombinasyonları doğrusal değildir. Bu yüzden doğrusal olmayan çoğu probleme karşı doğru bir tercih olacaktır. Sigmoid fonksiyonun avantajlarından biri de sonsuz değerler ile karşılaştığında her zaman (0, 1) aralığında değer üreteceği için aktivasyon değerini kaybetmez. Pürüzsüz eğrilere sahip olan sigmoid fonksiyonunda girdi değerlerindeki küçük değişiklikler çıktı değerlerinde gözlemlenebilir.

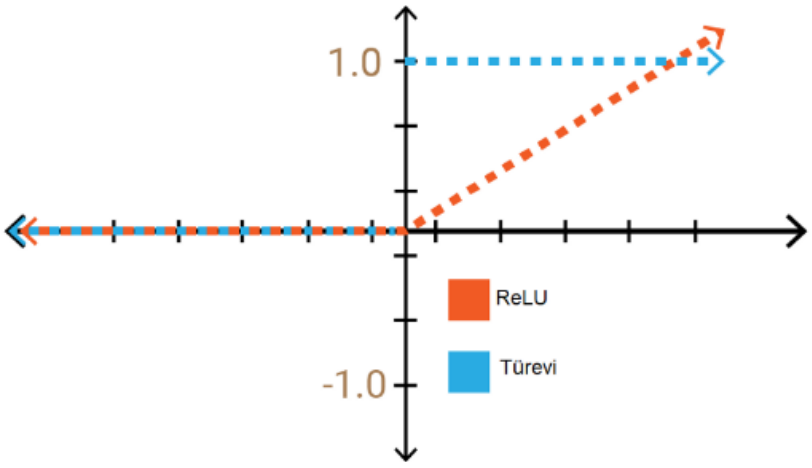
Hiperbolik Tanjant Fonksiyonu



Şekil 2.7: Hiperbolik Tanjant Fonksiyonu

Sigmoid fonksiyonuyla benzer bir yapıya sahiptir. Sigmoid fonksiyondan farklı olarak aralığı (-1, +1) olarak tanımlanır. Türevi daha dik olduğu için daha çok değer alabilir. Bu özelliğinden dolayı hızlı öğrenme ve sınıflandırma işlemleri için sigmoid fonksiyonuna göre daha verimli olacağı anlamına gelir.

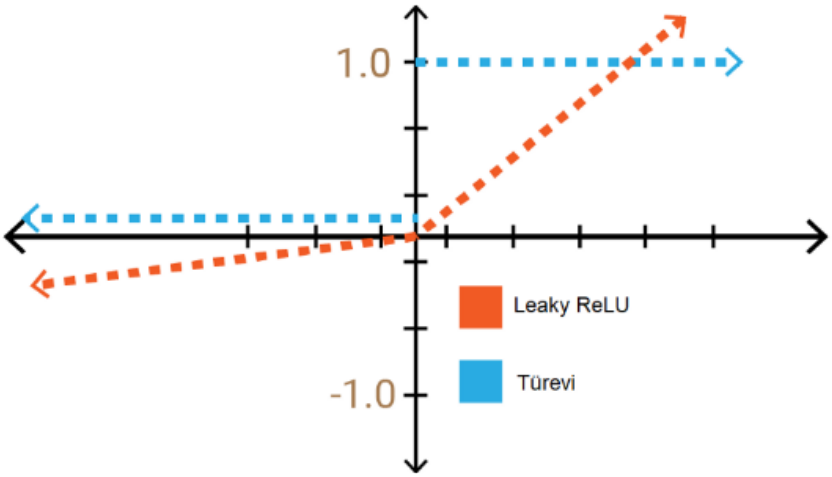
ReLU (Rectified Linear Unit) fonksiyonu



Şekil 2.8: ReLU (Rectified Linear Unit) Fonksiyonu

ReLU fonksiyonu [0, +∞) aralığında değer alır. ReLU fonksiyonun avantajı aynı anda tüm nöronları aktive etmemesidir. Yani eğer bir nöron negatif değerdeyse aktive edilmez ve 0 değerini alır. Nöronun 0 değerini alması ağın daha hızlı çalışacağı anlamına gelmektedir. Hesaplama yükünün az olması çok katmanlı ağlarda oldukça fazla tercih edilmesine yol açmıştır.

Sızıntı (Leaky) ReLU Fonksiyonu



Şekil 2.9: Sızıntı (Leaky) ReLU (Rectified Linear Unit) Fonksiyonu

Grafikten de görebileceğiniz gibi Leaky ReLU'da negatif değerler 0, 01 değeriyle sıfıra çok yakındır fakat tam sıfır değildir. Böylece türevinin sıfır olması engellenmiş olur. Böylece ReLU'da ki pasif gradyanları aktive ederek öğrenmeyi negatif bölgelerdeki değerler içinde sağlamış olur.

Softmax Fonksiyonu

Yapısal olarak sigmoid fonksiyonun benzerdir. Sınıflandırıcı olarak kullanıldığında oldukça iyi performans sergiler. Avantajı sigmoid fonksiyonunda olduğu gibi ikiden fazla sınıflandırma probleminde derin öğrenme modellerinin çıkış katmanında tercih edilir.

2.3.1.5 Çıktı

Aktivasyon fonksiyonu tarafından belirlenen çıktı değeridir. Bu değer yapay sinir ağındaki başka bir sinir hücresine girdi olarak sağlanabileceği gibi yapay sinir ağının sonucunu belirleyen son değerde olabilir.

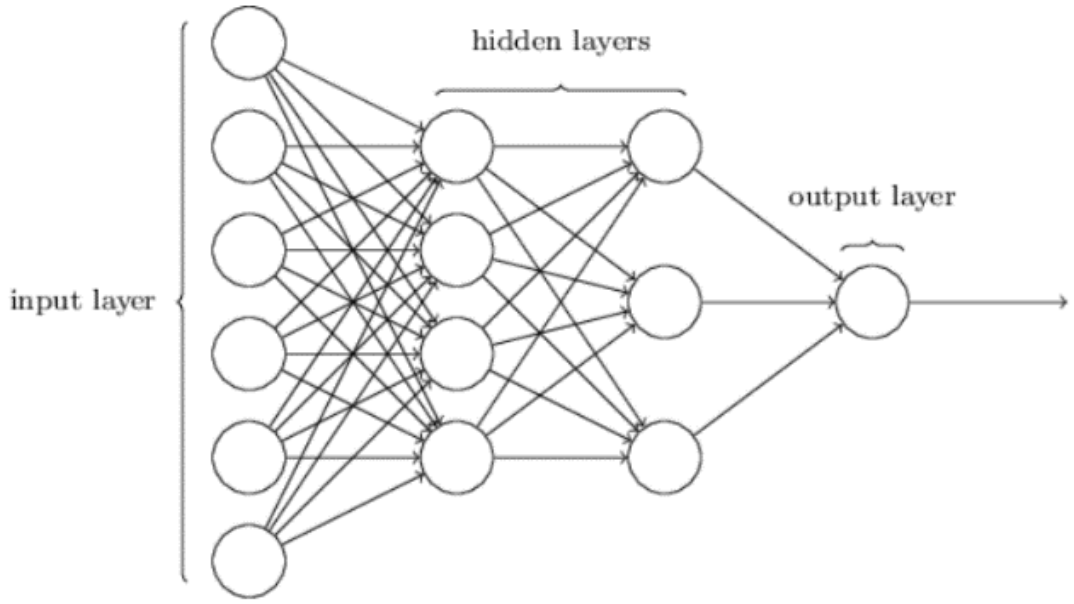
2.3.2 Yapay Sinir Ağı Modelleri

2.3.2.1 Tek Katmanlı Yapay Sinir Ağları

Sadece girdi ve çıktı katmanından oluşan, doğrusal problemlerin modellenmesinde başarılı olurken doğrusal olmayan karmaşık problemlerin çözümünde başarılı olamayan sinir ağlarıdır.

2.3.2.2 Çok Katmanlı Yapay Sinir Ağları

Yapısında en az 1 gizli katman içeren, karmaşık yani doğrusal olmayan problemlerin çözümünde başarılı olan sinir ağlarıdır. Çalışmamızda da kalp krizi riski sınıflandırma problemi için çok katmanlı yapay sinir ağları kullanılacaktır. Çok katmanlı yapay sinir ağlarının tek katmanlı sinir ağlarına kıyasla dezavantajı eğitim maliyeti kıyasla daha fazladır.



Şekil 2.10: Çok Katmanlı Yapay Sinir Ağı Örneği

Girdi Katmanı: Dışarıdan girdi (input) değerlerini alan nöronları içerir.
Gizli Katmanı: Girdilerin yapay sinir ağının parametrelerine göre çeşitli işlemlere tabi tutulduğu alandır. Gizli katman en az 1 katmandan oluşabileceği gibi daha fazla katmandan da oluşabilir.
Çıktı Katmanı: Bu katmandaki yapay sinir hücreleri, gizli katmanlardan gelen bilgileri işleyerek üretmesi gereken çıktıyı üreten katmanlardır.

2.3.2.3 İleri Beslemeli Yapay Sinir Ağları

Bu yapıdaki sinir ağlarında nöronlar girişten çıkışa kadar düzenli katmanlar şeklindedir. Bir katmanın yalnızca kendisinden sonraki katmanlara bağı bulunmaktadır.

2.3.2.4 Geri Beslemeli Yapay Sinir Ağları

İleri beslemeli ağlardan farkı bir nöronun çıktısı kendisinden sonraki nöron katmanına girdi olarak verilmemesidir. Yalnızca kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir nörona girdi olarak bağlanabilir.

2.3.3 Yapay Sinir Ağlarının Avantajları

2.3.3.1 Doğrusal Olmayan Yapı

Yapay sinir ağları gerçek hayatta karşılaşılan ve doğrusal olmayan yapıları da dikkate alabilirler. Örneğin özellikle finans sektöründe ekonomik veriler doğrusal olmayan bir yapıda ilerler. Ancak tahminlemeler lineer yöntemlerle kullanıldığında doğru sonucu olma olasılığı oldukça düşebilir. Yapay sinir ağları bu gibi örneklerde doğrusal olamayan yapıları da dikkate alabildiğinden oldukça avantajlı konumdadır.

2.3.3.2 Paralellik

Seri işlemcilerde herhangi birimin yavaşlığı tüm sisteme etki ederken yapay sinir ağlarının parelellik özelliğinden dolayı herhangi birimin yavaş olması tüm sinir ağını etkilemez.

2.3.3.3 Öğrenme

İnsan zihnini esas alan yapay sinir ağları eğitme veya başlangıç tecrübesi sayesinde veriyi kullanarak anlamlar çıkartıp öğrenme tecrübesine sahiptir. Bu özelliğiyle çok karmaşık problemlere çözüm üretebilmektedir.

2.3.3.4 Dağıtılmış Hafıza

Yapay sinir ağlarında bilgi, ağa yayılmış halde bulunmaktadır. Hücreleri arasındaki bağlantının değerleri ağın bilgisini de yansıtmaktadır. Tek bir bağlantının bir anlamı yokken ağın tamamı olayın bütünü için bir anlam ifade etmektedir.

2.3.3.5 Gerçek Zamanlı İşlem Yapabilme

Yapay sinir ağları hesaplamaları paralellik özelliğinden dolayı paralel olarak sürdürebildiğinden dolayı gerçek zamanlı işlem yapabilirler.

2.3.3.6 Genelleme

Yapay sinir ağları, öğrenme kabiliyeti sayesinde daha öncesinden bilinen örnekleri kullanarak daha önce karşılaşılmamış durumlar için genelleme yapabilmektedirler.

2.3.3.7 Kendi İlişkisini Oluşturma

Yapay sinir ağları verilere göre kendi içerisinde bir ilişki oluşturur. Bu ilişki herhangi bir paterne sahip olmadığı için doğru analiz edilemez. Bu özelliğinden dolayı 'akıllı karakutu' olarak isimlendirilir.

2.3.3.8 Sınırsız Sayıda Değişken ve Parametre

Yapay sinir ağları modelleri sınırsız sayıda değişken ve parametre ile çalışabildiklerinden dolayı harika bir öngörü doğruluğu ile çözümler sağlayabilmektedir.

2.3.3.9 Kademeli Bozulma

Ağlar herhangi bir problem ile karşılaştıkların hemen bozulmazlar. Hata payına sahip oldukları için kademeli bir şekilde tüm ağ bozulur [10].

2.3.4 Yapay Sinir Ağlarının Kullanım Alanları

Yapay Sinir Ağları günümüzde artık pek çok sektörde kendisine kullanım alanı bulmuştur.

- Tıbbi görüntü sınıflandırması yoluyla tıbbi tanılama
- Sosyal ağ filtreleme ve davranışsal veri analizi aracılığıyla hedeflenen pazarlama
- Finansal araçların geçmiş verilerini işleyerek finansal tahminlerde bulunma
- Elektrik yükü ve enerji talebi tahmini
- Süreç ve kalite kontrolü
- Kimyasal bileşik tanımlama

Yapay sinir ağlarının dört önemli uygulama alanı ise şöyle sıralanır:

2.3.4.1 Görüntü İşleme

Bilgisayarlar, sinir ağlarının avantajıyla nesnelerin tanımı yapıp ayırt edebilir duruma gelmişlerdir.

2.3.4.2 Konuşma Tanıma

Sinir ağları, dinamik olan konuşma kalıplarına, tonuna, dile ve aksan parametrelerine rağmen insan konuşmasını analiz edebilir.

2.3.4.3 Doğal Dil İşleme

Sinir ağları, bilgisayarların metin verilerinde ve belgelerdeki öngörüleri ve anlamı çıkarmasına yardımcı olur. Chatbotlar, spam maillerin tanınması gibi uygulamalar bu alanın örnekleridir.

2.3.5 Convolution Neural Network (CNN)

CNN genellikle görüntü işlemede kullanılan ve girdi olarak görselleri alan bir derin öğrenme algoritmasıdır. Farklı operasyonlarla görsellerdeki featureları (özellikleri) yakalayan ve onları sınıflandıran bu algoritma farklı katmanlardan oluşmaktadır. Convolutional Layer (Evrişimsel katman), Pooling ve Fully Connected olan bu katmanlardan geçen görsel, farklı işlemlere tabii tutularak derin öğrenme modeline hazır bir hale getirir [11].

2.3.5.1 Convolution Layer (Evrişimsel Katman)

Evrişimsel katman, bir CNN'nin temel yapı taşıdır ve hesaplamanın çoğunluğunun gerçekleştiği yerdir. Girdiler, filtre ve öznitelik haritası gibi birkaç bileşen gerektirir. Girişin 3 boyutlu piksel matrisinden oluşan renkli bir görüntü olacağını varsayalım. Bu, girdinin görüntüdeki RGB'ye karşılık gelen üç boyuta (yükseklik, genişlik ve derinlik) sahip olacağı anlamına gelir. Ayrıca, görüntünün alıcı alanları boyunca hareket ederek özelliğin mevcut olup olmadığını kontrol edecek bir özellik algılayıcı(feature detector) yani kernel veya filtre bulunmaktadır. Bu süreç genel anlamıyla convolution olarak bilinir.

Özellik algılayıcı, görüntünün bir kısmını temsil eden iki boyutlu (2 boyutlu) bir ağırlık dizisidir. Boyutları farklılık gösterse de filtre boyutu genellikle 3x3'lük bir matris olmaktadır; bu aynı zamanda alıcı alanın boyutunu da belirler. Daha sonra filtre görüntünün bir alanına uygulanır ve girdi pikselleri ile filtre arasında bir nokta çarpımı hesaplanır. Bu nokta çarpımı daha sonra bir çıktı dizisine beslenir. Daha sonra filtre yavaş yavaş kayar ve çekirdek görüntünün tamamını tarayana kadar işlemi tekrarlar. Girdi ve filtreden gelen nokta çarpım serisinin son çıktısı, öznitelik haritası veya aktivasyon haritası olarakta bilinir.
Bu işlemler yapılırken özellik algılayıcıdaki ağırlıkların görüntü boyunca hareket ettiği sürece sabit kalmaktadır. Ağırlık değerleri gibi bazı parametreler, eğitim sırasında geri yayılım(backprogation) aracılığıyla ayarlanır. Ancak yapay sinir ağı öğrenmeye başlamadan önce çıkışı etkileyen ve ayarlanması gereken üç önemli parametre vardır [12]. Bunlar;

1. **Filtre sayısı** çıktının derinliğini etkiler. Örneğin, üç farklı filtre, üç farklı öznitelik haritası üreterek üç derinlik oluşturur.

2. **Adım (stride)**, kernelin giriş matrisi üzerinde hareket ettiği mesafe veya piksel sayısıdır. İki veya daha büyük adım değerleri nadir olmakla birlikte, daha büyük bir adım daha küçük bir çıktı sağlar.

3. **Zero padding** genellikle filtreler giriş görüntüsüne uymadığında kullanılır. Bu, giriş matrisinin dışında kalan tüm öğeleri sıfıra ayarlayarak daha büyük veya eşit boyutlu bir çıktı üretir. Üç çeşit padding vardır:

- **Valid pedding:** Bu aynı zamanda no padding olarak da bilinir. Bu durumda boyutlar hizalanmazsa son convolution düşürülür.
- **Same padding:** Çıktı katmanının girdi katmanıyla aynı boyuta sahip olmasını sağlar
- **Full padding:** Girdi kenarlarına sıfırlar ekleyerek çıktının boyutunu artırır.

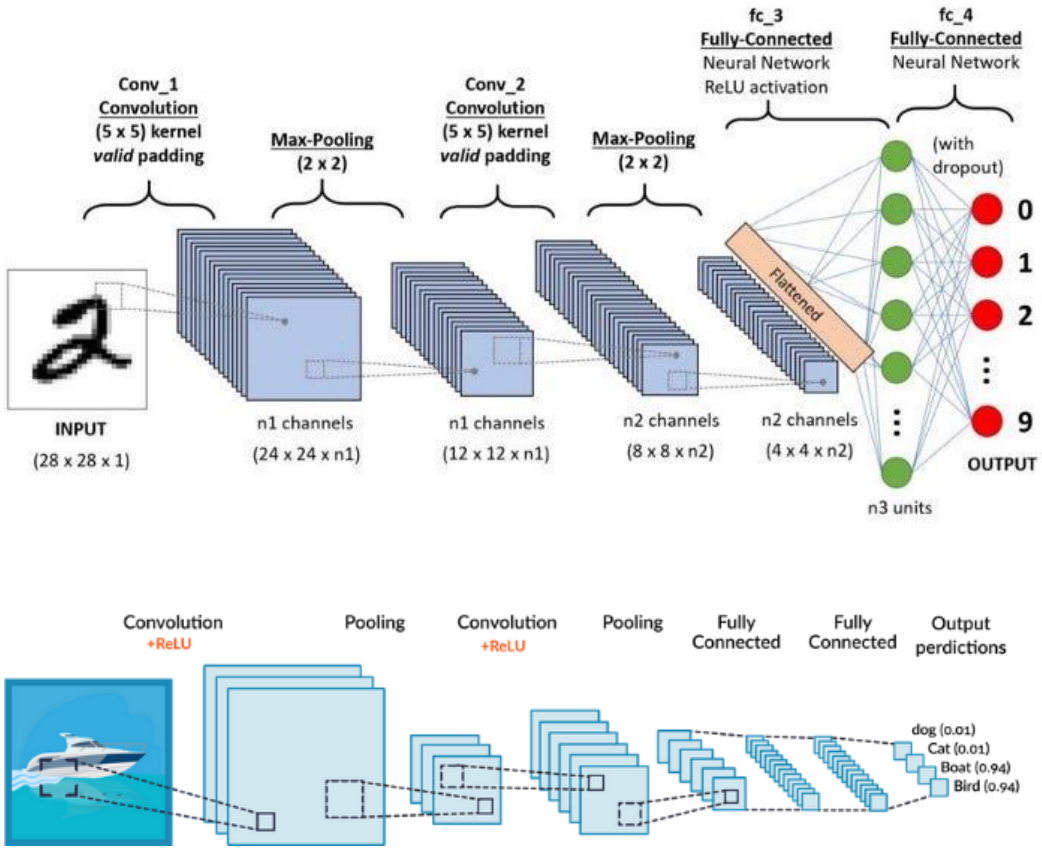
Her convolution işleminden sonra CNN, öznitelik haritasına bir Rectified Linear Unit (ReLU) dönüşümü uygulayarak modele doğrusal olmama (nonlinearity) özelliğini getirir.

2.3.5.2 Pooling Layer

Girdideki parametre sayısını azaltarak boyun azaltılmasını sağlar. Evrişim katmanına benzer şekilde, pooling işlemi, gidinin tamamı boyunca bir filtreyi tarar, ancak convolution katmanından farkı ise bu filtrenin herhangi bir ağırlığa sahip olmamasıdır. İki ana havuzlama türü vardır:

- **Max pooling:** Filtre girdi boyunca hareket ettikçe, çıktı dizisine gönderilecek maksimum değere sahip pikseli seçer. Bu yöntem average pooling'e göre daha sık kullanılmaktadır.
- **Average pooling:** Filtre girdi boyunca hareket ettikçe, çıktı dizisine gönderilecek alıcı (receptive) alan içindeki ortalama değeri hesaplar.

Pooling katmanında çok fazla bilgi kaybı yaşanabilirken, CNN'e de birçok faydası vardır. Karmaşıklığın azaltılmasına, verimliliğin artırılmasına ve overfit riskinin azaltılmasına yardımcı olurlar.



Şekil 2.11: Convolution Neural Network (CNN)

2.4 Derin Öğrenme Kütüphaneleri

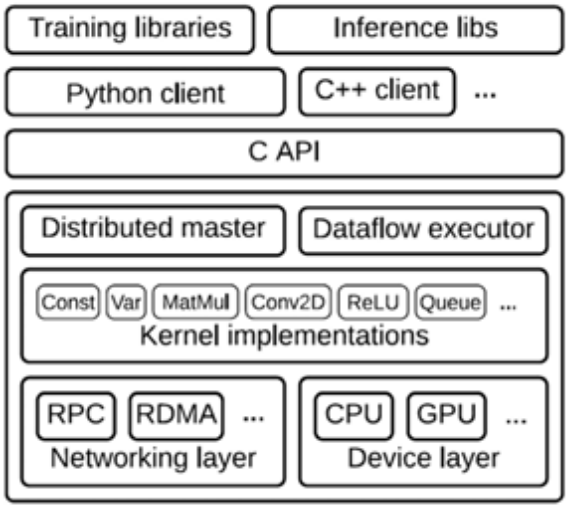
Derin Öğrenme uygulamaları Makine Öğrenmesinin bir alt kümesidir. Makine öğrenmesi ve derin öğrenme uygulamalarını geliştirmek için birçok uygulama çerçevesi ortaya çıkmıştır. Bir problem genellikle bu çerçevelerden bir tanesiyle çözülememektedir. Kütüphanelerin birden fazlasının bir arada kullanılmasıyla çözüme gidilmektedir [13]. Şekil 2.12'de farklı makine öğrenmesi araçlarının sınıflandırması yer almaktadır.

Şekil 2.12: Farklı makine öğrenmesi araçları [13]

Derin öğrenmenin popülerliği her geçen gün atmaktadır. Buna paralel olarak faklı derin öğrenme çerçeveleri geliştirilmektedir. TensorFlow (2014), Keras (2014), Theano (2008), caffe (2012), CNTK (2016) bunlara örnek olarak gösterilebilir [14], [15]. Bu derin öğrenme kütüphaneleri Python, c++, java gibi farklı programlama dilleri temel alınarak geliştirilmiştir [16].

2.4.1 TensorFlow

TensorFlow Makine öğrenmesi ve derin öğrenme projelerini baştan sona tasarlayıp sonuçları görebileceğimiz bir ortam sunar [17]. Google Brain ekibi tarafından 2015 yılında açık kaynaklı bir proje olarak yayınlanmıştır. Birbirinde farklı ortamlarda çalışabilir. Farklı programlama dilleri ve farklı donanımlarda çalışabilir. Bu kütüphanedeki yapı temel olarak bir dizi hesaplamalardan meydana gelen veri akış grafiklerinden oluşmaktadır. Bu akış grafikleri düğümlerin durumunu kaydetmek, güncellemek için dallanma ve döngü kontrolüne izin veren bir veri akışı hesaplaması sunar [18]. TensorFlow, otomatik farklılaştırma ve parametre paylaşma yetenekleri nedeniyle farklı mimari türlerini destekler. TensorFlow, paylaşılan parametreleri güncellemek için iş birliği yapan çoklu hesaplama kaynaklarını kullanarak veri akış grafiği modelinin paralel yürütülmesi yoluyla paralelliği destekler. Paralel çalışma yeteneği ciddi performans artışlarına izin verir. TensorFlow ayrıca mobil ve Nesnelerin İnterneti (IoT) cihazlar için optimize edilmiş çıkarım yapma sürümünü de sunmaktadır. Düşük donanım özelliklerinde etkin sonuçlar vermesi hedeflenmiştir [19]. TensorFlow ile Google ve Amazon gibi bulut ortamlarında model oluşturmak, eğitmek ve test etmek mümkündür. Şekil 2.13'de TensorFlow mimari katmanlarına yer verilmiştir.



Şekil 2.13: TensorFlow mimari katmanlarının şematik gösterimi [18]

2.4.2 Keras

Keras, Python'da geliştirilmiş bir derin öğrenme kütüphanesidir. Derin öğrenme modellerinin geliştirilmesi eğitim ve test edilmesi görevlerini yerine getirir [20]. Kullanıcılara yönelik daha kolay araçlar geliştirmeye odaklanmıştır bu nedenle kullanıcı dostudur. TensorFlow ve Theano üzerinde üst düzey uygulama programlama arayüzü (API)'ler sunar. 2021'in başlarında 400.000'den fazla bireysel kullanıcıyla Keras, hem endüstri hem de araştırma topluluğu genelinde güçlü bir şekilde benimsenmiştir [21]. Aynı kodun değişmeden hem merkezi işlem birimi (CPU), hem de grafik işlem birimi (GPU)'da çalışmasını sağlar. Derin öğrenme modellerinin hızlıca prototipleştirilmesini sağlayan API'ler sunar. Evrişimli ağlar, yinelemeli ağlar ve her ikisinin birlikte çalışabilmesi için önceden tanımlı araçlar sunar.

2.4.3 Theano

Theano, Montreal Üniversitesi'nden araştırmacılar ve geliştiriciler tarafından geliştirilmiştir. Theano, Berkeley Software Distribution (BSD) lisansı altında lisanslanan ücretsiz, açık kaynaklı bir yazılımdır. Dünya çapında geniş ve çok aktif bir geliştirici ve kullanıcı topluluğuna dayanır. Theano, derin öğrenme modelleri oluşturmayı kolaylaştıran temel bir matematiksel ifade kütüphanesidir. Theano, çok boyutlu dizileri içeren matematiksel ifadeleri verimli bir şekilde tanımlamaya, optimize etmeye ve değerlendirmeye izin veren bir Python kitaplığı olarak tanımlanabilir. Piyasaya sunulduğundan bu yana, özellikle makine öğrenimi topluluğunda çok fazla kullanılan CPU ve GPU matematiksel derleyicilerinden biri olmuştur [22]. Theano, birçok modern makine öğrenimi modelinde inşa edilen ve kullanılan çok sayıda üstyapı ile 2008'den beri aktif ve sürekli olarak geliştirildi. Ancak, Theono'nun geliştiricilerinden Yoshua Bengio, 2017'de Theano'daki geliştirmenin sona ereceğini duyurdu [23].

2.4.4 Caffe ve Caffe 2

Caffe, derin öğrenme algoritmalarını uygulamak için Kaliforniya Üniversitesi, Berkeley tarafından geliştirilen açık kaynaklı bir platformdur. Kodlar, GPU hesaplama için kullanılan Bilgisayar Birleşik Cihaz Mimarisi (CUDA) kütüphanesi ile C++'da

yazılır. Ayrıca CUDA kütüphanesi, Python/Numpy ve MATLAB'ı destekler. Caffe, ilk olarak görüntü işleme için tasarlanmasına rağmen daha sonradan ses tanıma, robotik, astronomi ve sinir bilimi için kullanılmıştır [24]. Derin Sinir Ağları (DNN), Caffe'de katman katman tanımlanır. Katman, bir modelin özü ve hesaplamanın temel birimidir. Veriler, Caffe'ye veri katmanları aracılığıyla girer. Kabul edilen veri kaynakları, verimli veri tabanları (LevelDB veya LMDB), Hiyerarşik Veri Formatı (HDF5) veya yaygın görüntü formatlarıdır (örn. GIF, TIFF, JPEG, PNG, PDF). Nisan 2017'de son kararlı sürümü yayımlanmıştır. Caffe2, Caffe'nin geliştiricisi Yangqing Jia tarafından geliştirilmiştir. Yangqing jia, Facebook'da çalışmaya başladıktan sonra, NVIDIA ve Facebook ile birlikte caffe tabanlı Caffe2 çerçevesini geliştirdiler. Caffe2, büyük ölçekli dağıtılmış eğitim desteği, mobil dağıtım, yeni donanım desteği nicelenmiş hesaplama gibi bazı Caffe sınırlılıklarını iyileştirdi. Caffe2, NVIDIA desteğine sahiptir. Ayrıca python ve c++ API desteği de mevcuttur.

2.4.5 Torch

Makine öğrenimi algoritmaları için destek sunan BSD lisanslı bilimsel hesap kütüphanesidir. Facebook, Twitter, Google gibi kuruluşlar tarafından desteklenmektedir. Hem CPU hem de GPU'da çalışabilir. Esas olarak büyük ölçekli öğrenme (konuşma, görüntü ve video uygulamaları), denetimli öğrenme, denetimsiz öğrenme, pekiştirmeli öğrenme, Sinir Ağları, optimizasyon, grafik modeller, görüntü işleme için kullanılır. Son sürüm olan Torch7 ile geliştirmesi durdurulmuştur.

2.4.6 PyTorch

PyTorch, Facebook'un AI araştırma grubu tarafından Ekim 2016'da tanıtıldı. PyTorch, API aracılığıyla derin öğrenme modelleri oluşturmayı kolaylaştıran Python tabanlı bir derin öğrenme çerçevesidir. Statik hesaplama grafikleri kullanan diğer popüler derin öğrenme çerçevelerinin çoğunun aksine PyTorch, karmaşık mimariler oluşturmada daha fazla esneklik sağlayan dinamik hesaplama kullanır [25]. PyTorch, birçok kütüphaneyle entegre çalışır. Büyük, küçük birçok sinir ağında yüksek performans sunar. PyTorch'daki bellek kullanımı, Torch veya bazı alternatiflere kıyasla son derece verimlidir. Derin öğrenme modellerinizin maksimum düzeyde bellek açısından verimli olmasını sağlamak için GPU'ya özel bellek ayırıcıları

mevcuttur. Buda, eskisinden daha büyük derin öğrenme modellerini eğitmenizi sağlar [26]. Kütüphane bir BSD lisansı altında ücretsiz olarak sunulur. Facebook, Twitter, NVIDIA ve diğer birçok kuruluş tarafından desteklenmektedir.

2.4.7 MXNet

MXNet, Carnegie Mellon Üniversitesi, Washington Üniversitesi ve Microsoft arasındaki iş birliği ile ortaya çıkmış açık kaynaklı bir derin öğrenme çerçevesidir. C, Python, MATLAB, JavaScript, R, Julia ve Scala dahil olmak üzere farklı programlama dillerini kullanarak derin sinir ağlarının eğitimine izin veren ölçeklenebilir bir çerçevedir. MXNet, çoklu CPU'larda veya GPU'larda veri paralelliğini destekler ve model paralelliğine de izin verir. MXNet, özellikle derin sinir ağları için makine öğrenimi algoritmalarının geliştirilmesini kolaylaştıran çok dilli bir makine öğrenimi (ML) kitaplığıdır. MXNet, hesaplama ve bellek açısından verimlidir ve mobil cihazlardan dağıtılmış GPU kümelerine kadar çeşitli heterojen sistemlerde çalışır [27].

Bölüm 3

Yöntem

Her dönem projesi, kapak, ön sayfalar, ana metin, kaynaklar ve ekler kısımlarını içerir; ayrıca başka kısımlar olabilir. Bu bölümde bunlardan sırasıyla bahsedilecektir.

3.1 Veri Seti

Bu çalışmadaki veri seti temel ve karmaşık duyguları kapsamaktadır. İnsanlar günlük hayatlarında birçok duyguyu tanısalar da çalışmalar genellikle 7 temel duyguyu esas almıştır. Kaggle.com websitesinde bulunan, 35.685 adet 48x48 pixel tek renk yüz ifadelerini içeren "Emotion Detection" verisetinin CNN derin öğrenme algoritması yardımıyla çeşitli yüz ifadelerini(mutluluk, nötr, üzüntü, öfke, şaşkınlık, tiksinti, korku)sınıflandıran bir modelin oluşturulması amaçlanmıştır. Bu sınıflandırma yapılırken Confusion matrix, accuracy score, ROC etc. değerlendirme ölçütleri

yardımıyla oluşturulan modelin değerlendirilmesi ve yüz ifadesi içeren herhangi bir fotoğrafın tahmininin yapılması hedeflenmektedir. Veri setinden örnek görüntüler aşağıda gösterilmektedir.

Etiketlerin gösterilmesi

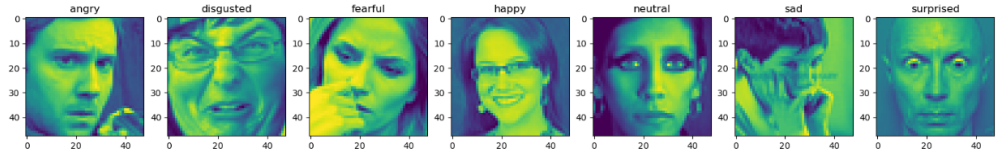
```
In [3]: end_path = os.listdir(train_path)
        end_path
Out[3]: ['angry', 'disgusted', 'fearful', 'happy', 'neutral', 'sad', 'surprised']
```

Şekil 3.1: Veri setindeki Etiketlerin gösterilmesi

7 adet farklı yüz ifadesinin 1 satırda ve 7 sütunda gösterilmesi

```
In [13]: import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 7, figsize=(20,8))
for i in range(len(end_path)):
    j = train_path+end_path[i]+'/'+os.listdir(train_path+end_path[i]+'/')[0]
    axes[i].imshow( plt.imread(j) )
    axes[i].set_title(end_path[i])
plt.show()
```



Şekil 3.2: Veri setindeki örnek görüntüler

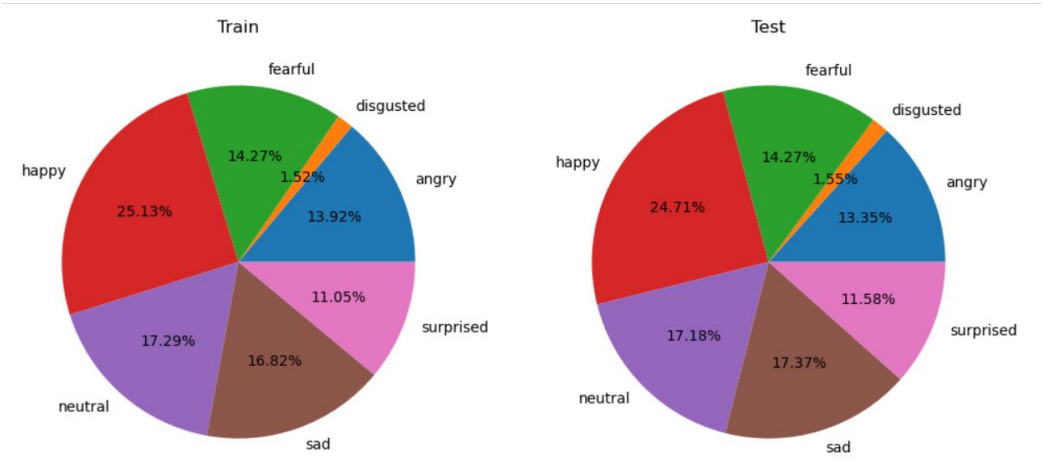
Veri setimizi incelediğimizde veri setinin esasında üzerinde bir model uygulanmasına elverişli yani hazır olduğu gözlemlenmektedir. Veri setimizin işlemeye açık olmasının nedenlerinden biri açık kaynaklı bir veri seti olmasından kaynaklanıyor. Eğer veri setimizi kendimiz hazırlıyor olsaydık ya da bu kadar işlenmiş bir veri seti kullanmıyor olunsaydı o zaman veri seti gerekli teknikler yardımıyla ön işlenmesi gerekirdi.

```
1 satır 2 sütündan oluşan pasta grafiklerinin oluşturulması
İlk pasta grafiğinde train setindeki yüz ifadelerinin yüzdesel olarak dağılımı bulunmaktadır.
İkinci pasta grafiğinde ise test setindeki yüz ifadelerinin yüzdesel olarak dağılımı bulunmaktadır.

In [26]: x_train = np.array([ len(os.listdir(train_path+i+'/')) for i in end_path ])
         x_test = np.array([ len(os.listdir(test_path+i+'/')) for i in end_path ])
         label = end_path

         fig, axes = plt.subplots(1, 2, figsize=(12,8))
         axes[0].pie(x_train, labels=label, autopct='%1.2f%%', startangle=0)
         axes[1].pie(x_test, labels=label, autopct='%1.2f%%', startangle=0)
         axes[0].set_title('Train')
         axes[1].set_title('Test')
         plt.show()
         from colorama import Fore, Back, Style
         for i in end_path:
             print(Fore.RED+'Emotion : ' + i )
             print(Fore.GREEN+'\tTrain : ' + str(len(os.listdir(train_path+i+'/'))) +'\n\tTest : ' + str(len(os.listdir(test_path+i+'/')))
```

Şekil 3.3: Veri Setindeki görüntülerin sayısal olarak ve pasta grafiğinde dağılımının oluşturulması



Şekil 3.4: Veri Setindeki görüntülerin pasta grafiğinde dağılımı

```
Emotion : angry
        Train : 3995
        Test : 958
Emotion : disgusted
        Train : 436
        Test : 111
Emotion : fearful
        Train : 4097
        Test : 1024
Emotion : happy
        Train : 7215
        Test : 1774
Emotion : neutral
        Train : 4965
        Test : 1233
Emotion : sad
        Train : 4830
        Test : 1247
Emotion : surprised
        Train : 3171
        Test : 831
```

Şekil 3.5: Veri Setindeki görüntülerin sayısal olarak dağılımı

3.2 Veri Önişleme

Veri seti üzerinde bir model eğitilmeden önce veri setinin olabildiğince tanımak ve eğitilecek modelin üzerinde doğru çalışacak halde olup olmadığı konusuna dikkat edilmelidir. Veri seti eğer üzerinde uygulanacak modellerin çalışmasına uygun değilse alacağımız sonuçlardan dolayı oldukça yanıltıcı olabilir.

Veri seti örnek, kolon sayıları gibi değerler ile tanıdıktan sonra veri setinin kolonlarında herhangi bir boşluk olup olmadığı tespit edilmelidir. Bir boşluk mevcutsa uygun değerler ile doldurulup algoritmalar ile işlemeye hazır hale getirilmelidir. Bu işlemler sayısal içerik olan verisetleri için geçerlidir. Bu çalışmadaki veriseti görsel

içeriklerden oluştuğu için böyle bir çalışma söz konusu değildir. Fakat bu projede doğrulama sayılarını train testinin yüzde 30'una denk gelecek kadarı seçildi.

Veriyi eğitmek için doğrulama sayısı train verisetinin yüzde 30' u olarak ayarlanmıştır.

```
In [28]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
         train_idg = ImageDataGenerator(rescale=1./255, validation_split=0.3)
         test_idg = ImageDataGenerator(rescale=1./255)

         img_size = (48, 48)
         batch_size = 64

         arg_train = {'target_size': img_size,
                      'color_mode': 'grayscale',
                      'class_mode' : 'categorical',
                      'batch_size': batch_size}
         arg_test = {'target_size': img_size,
                     'color_mode': 'grayscale',
                     'class_mode' : 'categorical',
                     'batch_size': batch_size,
                     'shuffle': False}

         train = train_idg.flow_from_directory(directory=train_path, subset='training', **arg_train)
         valid = train_idg.flow_from_directory(directory=train_path, subset='validation', **arg_train)
         test  = test_idg.flow_from_directory(directory=test_path, **arg_test)

         Found 20099 images belonging to 7 classes.
         Found 8610 images belonging to 7 classes.
         Found 7178 images belonging to 7 classes.
```

Şekil 3.6: Doğrulama sayılarının oluşturulması

Modelin compile edilmesi. Birden fazla kategori sayısı mevcut olduğu için kayıp fonsiyonunu categorical_crossentropy kullanmak daha mantıklıdır.

```
In [9]: model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0005),loss='categorical_crossentropy',metrics=['accuracy'])
        history = model.fit(
            train ,
            validation_data=valid,
            epochs=100)
```

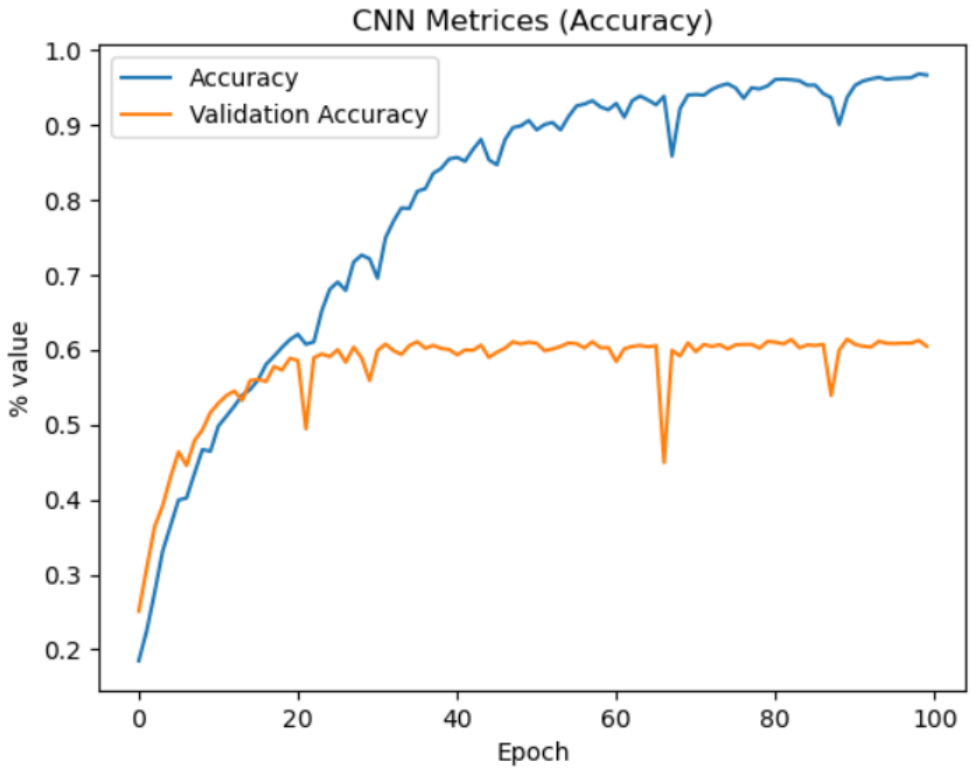
Şekil 3.7: Modelin Compile Edilmesi

Doğruya daha yakın tahminlere ulaşmak için epoch sayısı 100 seçildi. Çıktıda birden fazla kategori sayısı olduğu için kayıp fonksiyonu categorical_crossentropy olarak belirlendi.

Bölüm 4

Sonuç

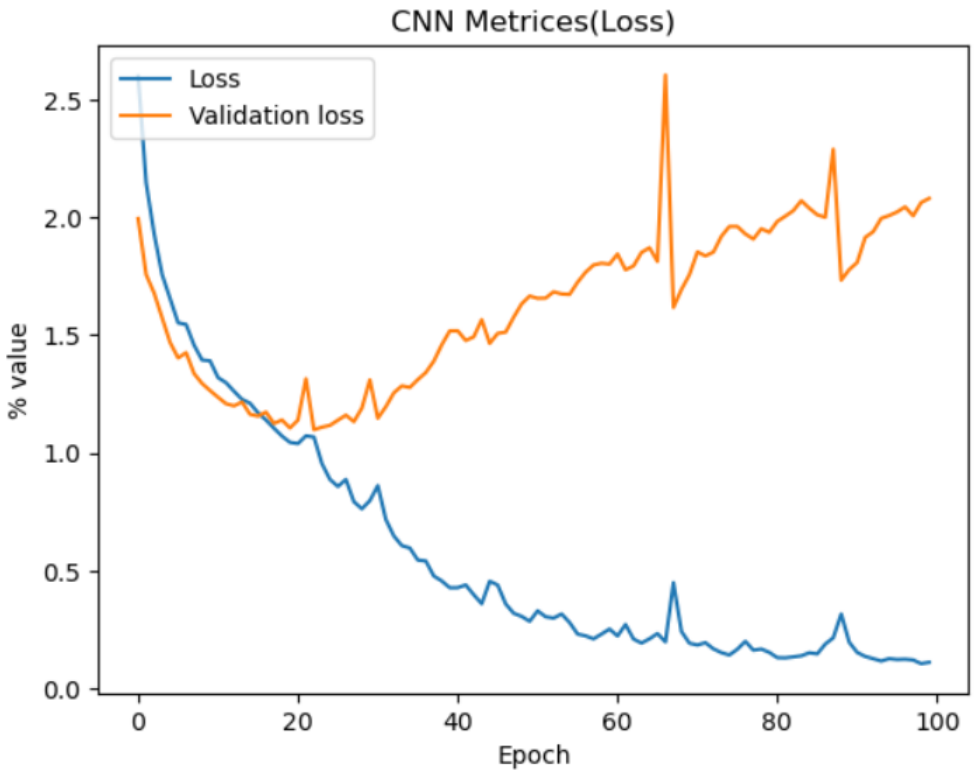
CNN Derin öğrenmesi algoritmasıyla oluşturduğumuz çok katmanlı modelin veriseti üzerindeki sonuçları aşağıdaki gibidir.



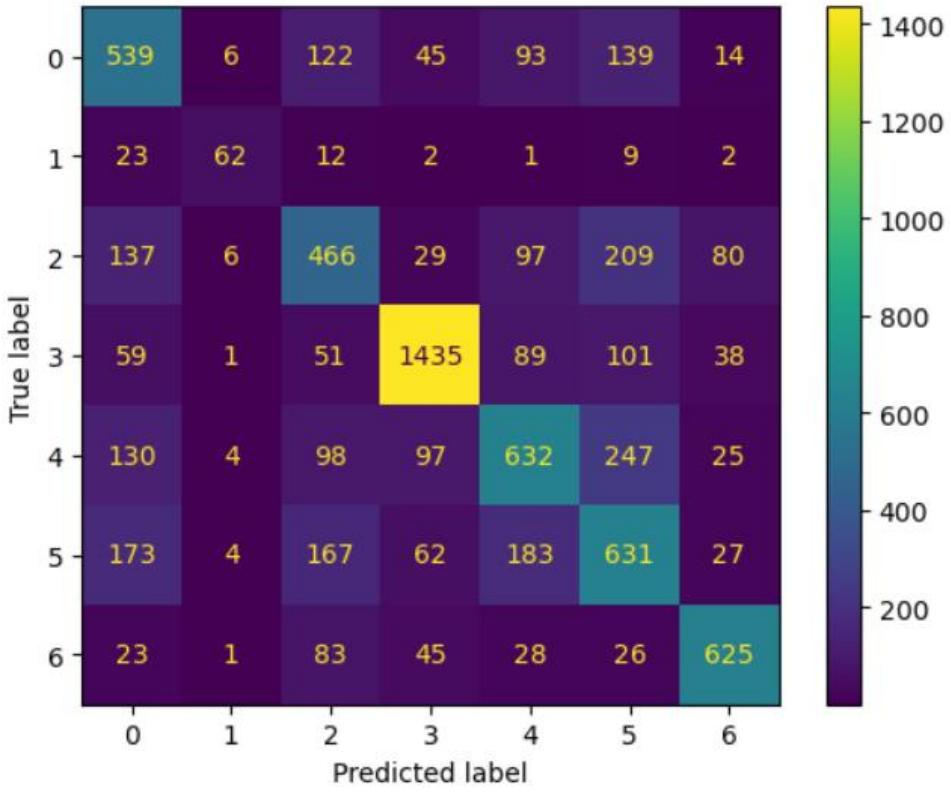
Şekil 4.1: CNN Metrices (Accuracy)

Grafikten anlaşılacağı üzere doğruluk değerinin yaklaşık olarak 40 Epoch'tan sonra neredeyse sabit kaldığı götülmektedir.

Bir diğer grafikte ise loss değerinin 40-45 Epoch'tan sonra hemen hemen değişmediği yorumlanabilmektedir.



Şekil 4.2: CNN Metrices (Loss)



Şekil 4.3: Confusion Matrix

	precision	recall	f1-score	support
0	0.4972	0.5626	0.5279	958
1	0.7381	0.5586	0.6359	111
2	0.4665	0.4551	0.4607	1024
3	0.8367	0.8089	0.8226	1774
4	0.5628	0.5126	0.5365	1233
5	0.4633	0.5060	0.4837	1247
6	0.7707	0.7521	0.7613	831
accuracy			0.6116	7178
macro avg	0.6193	0.5937	0.6041	7178
weighted avg	0.6175	0.6116	0.6136	7178

Şekil 4.4: Classification Report

Sonuç olarak elimizde bulunan yüz ifadesi örneklerimizden bir tanesine, oluşturduğumuz modeli uyguladığımızda modelimizin başarılı bir şekilde çalıştığını görmekteyiz.

In [52]:
```
image_path = 'C:/Users/kapcak/Desktop/bitirme projesi/test/angry/im1.png'
image = Image.open(image_path)

# Preprocess the image
img = image.resize((48, 48))
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

# Make predictions
predictions = loaded_model.predict(img_array)
class_labels = end_path
score = tf.nn.softmax(predictions[0])
print(f"{class_labels[tf.argmax(score)]}")
```

```
1/1 [==============================] - 0s 16ms/step
angry
```

Şekil 4.5: Tahmin Oluşturulması

Kaynaklar

[1] C.-T. Liao, H.-J. Chuang, C.-H. Duan, and S.-H. Lai, “Learning spatial weighting for facial expression analysis via constrained quadratic programming,” Pattern Recognit., vol. 46, no. 11, pp. 3103–3116, Nov. 2013.

[2] S. Tripathi, S. Acharya, R.D. Sharma, S. Mittal, S. Bhattacharya Using Deep and Convolutional Neural Networks for Accurate Emotion Classification on DEAP Dataset

[3] Chen, M., Zhang, L., & Allebach, J.P. (2015). Learning deep features for image emotion classification. 2015 IEEE International Conference on Image Processing (ICIP), 4491-4495.

[4] A. Sepas-Moghaddam, A. Etemad, P. L. Correia, and F. Pereira, “A Deep Framework for Facial Emotion Recognition using Light Field Images,” in 2019 8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019, 2019, pp. 482– 488.

[5] M. M. Taghi Zadeh, M. Imani, and B. Majidi, “Fast Facial emotion recognition Using Convolutional Neural Networks and Gabor Filters,” in 2019 IEEE 5th Conference on Knowledge Ba

[6] Nabiyev, V. V. (2012). Yapay Zekâ: İnsan-Bilgisayar Etkileşimi. Baski Yeri: Seçkin Yayıncılık.

[7] Andrew, A. M., (1991). Artificial Intelligence. Boston: AddisonWesley Company

[8] Popov, E. V., (Ed), (1990). Yapay Zekâ. Uzman Sitemler ve Doğal Dil İşleme. Moskova: Radio i Svyaz, s. 461.

[9] Öztemel, E. (2003). Yapay Sinir Ağları. İstanbul: Papatya, s.15-18.

[10] 2003: Yapay Sinir Ağları. Papatya Yayınevi, İstanbul.

[11] https://bartubozkurt35.medium.com/cnn-convolutional-neural-networks-nedir-a5bafc4a82a1

[12] https://www.ibm.com/topics/convolutional-neural-networks

[13] G. Nguyen et al., "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," Artif. Intell. Rev., vol. 52, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z.

[14] A. T. KABAKUŞ, "A Comparison of the State-of-the-Art Deep Learning Platforms: An Experimental Study," Sak. Univ. J. Comput. Inf. Sci., vol. 3, no. 3, pp. 169–182, Sep. 2020, doi: 10.35377/saucis.03.03.776573.

[15] K. Dinghofer and F. Hartung, "Analysis of Criteria for the Selection of Machine Learning Frameworks," in 2020 International Conference on Computing, Networking and Communications, ICNC 2020, Feb. 2020, pp. 373–377, doi: 10.1109/ICNC47757.2020.9049650.

[16] J. Liu, Q. Huang, X. Xia, E. Shihab, D. Lo, and S. Li, "Is Using Deep Learning Frameworks Free? Characterizing Technical Debt in Deep Learning Frameworks," in 2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS), 2020, pp. 1–10.

[17] "Neden TensorFlow." https://www.tensorflow.org/about?hl=tr (accessed May 19, 2021).

[18] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," 2016.

[19] "TensorFlow Lite | Mobil ve Uç Cihazlar için Makine Öğrenimi." https://www.tensorflow.org/lite/?hl=tr (accessed May 19, 2021).

[20] F. Chollet, Deep Learning with Phyton. 2018.

[21] "Neden Keras'ı seçmelisiniz?" https://keras.io/why_keras/ (accessed May 19, 2021).

[22] The Theano Development Team et al., “Theano: A Python framework for fast computation of mathematical expressions,” May 2016, Accessed: May 19, 2021. [Online]. Available: http://arxiv.org/abs/1605.02688.

[23] M. M. Yapıcı and N. Topaloğlu, “Performance comparison of deep learning frameworks,” 2021. Accessed: May 18, 2021. [Online]. Available: https://dergipark.org.tr/tr/pub/ci.

[24] A. Uçar, Ö. H. Üniversitesi, and M. Bölümü, “Derin öğrenmenin Caffe kullanılarak grafik işleme kartlarında değerlendirilmesi Mehmet Safa BİNGÖL,” 2018.

[25] R. Elshawi, A. Wahab, A. Barnawi, and S. Sakr, “DLBench: a comprehensive experimental evaluation of deep learning frameworks,” Clust. Comput. J. Networks, Softw. Tools Appl., p. 1, 2021, doi: 10.1007/s10586-021-03240-4.

[26] “pytorch/pytorch: Tensors and Dynamic neural networks in Python with strong GPU acceleration.” https://github.com/pytorch/pytorch (accessed May 20, 2021).

[27] T. Chen et al., “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems,” Dec. 2015, Accessed: May 20, 2021. [Online].