T.C.
KASTAMONU ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ I/II

KONU

Yüz Tanıma ile Duygu Durumu Belirleme

HAZIRLAYAN

184410055 – Tansu YILMAZ

DANIŞMAN

Doç. Dr. Salih GÖRGÜNOĞLU

Temmuz - 2020
KASTAMONU

T.C.
KASTAMONU ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ I/II

KONU

Yüz Tanıma ile Duygu Durumu Belirleme

HAZIRLAYAN

184410055 – Tansu YILMAZ

DANIŞMAN

Doç. Dr. Salih GÖRGÜNOĞLU

Temmuz - 2020
KASTAMONU

…/…/ 2020

ETİK BEYAN

Kastamonu Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, Mühendislik Tamamlama Programı, Tez Hazırlama Kılavuzu'nda yer alan kurallara uygun olarak hazırladığım bu çalışmada; proje 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, proje çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu projede sunduğum çalışmanın özgün olduğunu, bildirir; aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Öğrenci Numarası **:** 184410055 **İmza**

Adı Soyadı **:** Tansu YILMAZ

ÖZET

Bitime Projesi

DUYGU TANIMA

Tansu YILMAZ

Kastamonu Üniversitesi
Bilgisayar Mühendisliği Bölümü

Bitirme Projesi Danışmanı:
Doç. Dr. Salih GÖRGÜNOĞLU
Temmuz 2020, 35 sayfa

Önemi ve kullanımı gün geçtikçe artan yüz tanıma uygulamaları, derin öğrenme, yapay zeka bileşenleri yardımıyla, insanların duygularını tahmin edebilen bir yazılım gerçekleştirilmesi amaçlanmıştır. Yazılım çalıştırıldığında kamera programı çalışacak, ekrana bakan yüz tespit edilecek ve sahip olduğu duygu durumu tahmin edilmeye çalışılacaktır. Bu yazılımın oluşturulması sürecinde kullanılacak olan yöntem teknik ve uygulamalar konusunda bilgi verilmiştir. Anaconda yazılımı, OpenCV kütüphanesi, Arayüz yazılımları, Haar Cascade Sınıflandırıcı, Fer2013 Veri Kümesi, CNN Konvolüsyonel Sinir Ağları hakkında derlemeler yapılmıştır.

Anahtar Sözcükler : Duygu tanıma, Yapay zeka, Derin öğrenme, OpenCV , Visual Studio Code, Anaconda, Fer2013 Veri Kümesi, CNN Konvolüsyonel Sinir Ağları, Haar Cascade Sınıflandırıcı.

ABSTRACT

Senior Project

EMOTION RECOGNITION

Tansu YILMAZ

Kastamonu University
Faculty of Engineering and Architecture
Department of Computer Engineering

Project Advisor:
Assoc. Dr. Salih GÖRGÜNOĞLU
July 2020, 35 pages

It is aimed to realize a software that can predict the feelings of people with the help of facial recognition applications, deep learning and artificial intelligence components whose importance and usage are increasing day by day. When the software is run, the camera program will run, the face facing the screen will be detected and the emotional state it has will be tried to be estimated. Information on the methods, techniques and applications to be used in the process of creating this software is given. Compositions were made about Anaconda software, OpenCV library, Interface software, Haar Cascade Classifier, Fer2013 Data Set, CNN Conventional Neural Networks.

Key Words : Emotion Reconhnition, Artificial Intelligence,Deep Learning, OpenCV , Visual Studio Code, Anaconda, Fer2013 Dataset, CNN Conventional Neural Network, Haar Cascade Classifier.

İÇİNDEKİLER **SAYFA**

1. GİRİŞ ve TANITIM

Yüz tanıma, yapay zeka çalışmalarında oldukça popüler olan ve pratik uygulamalardaki önemi gün geçtikçe artan bir çalışma alanıdır. En yalın haliyle yüz tanıma, önceden sisteme tanıtılan insanların sistem tarafından daha sonra otomatik olarak tanınmasını amaçlamaktadır. Bilgisayarla görü, görüntü işleme ve örüntü tanıma gibi temel bilgisayar bilimleri dallarıyla ilişkili olan yüz tanımanın, yüzlerin sabit fotoğraflar veya hareketli görüntülerden alınması, görüntü alındığı esnada yüzlerin pozu veya açısı, ortamın ışığı, yüzlerde sakal/bıyık benzeri doğal farklılaştırıcılar veya gözlük, atkı gibi aksesuarların bulunması gibi birçok etkene bağlı olmasından dolayı her bir etkeni ele almaya yönelik farklı teknikler ve yöntemler geliştirilmiştir.

Günümüzde güvenlik uygulamalarında kullanımı yaygınlaşmış olan ve biyometrik fotoğraf olarak anılan, belli özel koşullar altında çekilen vesikalık fotoğraflar gibi şartların daha önceden sabitlendiği ve yüzün tüm tanımlayıcı öğelerinin (göz, burun, ağız vb.) net bir şekilde erişilebilir olduğu fotoğraflardan yüz tanımaya karşın hareketli görüntülerden yüz tanıma halen daha, bir çok zorlukla karşılaşılan bir uygulamadır. Çünkü bir önceki paragrafta bahsedilen neredeyse tüm etkenlerin karşılaşıldığı hareketli görüntülerden yüz tanıma, bu zorluklara karşın güvenlikten eğlenceye kadar bir çok konuda uygulama alanı bulabilmektedir. Örneğin kameralardan çekilen güvenlik kayıtlarındaki kişilerin otomatik olarak tanınması, bir filmdeki oyuncuların otomatik olarak listelenmesi veya insan-makine etkileşimi gibi uygulamalar temelde hep hareketli görüntülerden yüz tanımaya dayanır.

Kaynak ne olursa olsun (video, canlı kayıt, gerçek zamanlı görüntü vs.) hareketli görüntülerden yüz tanıma temelde iki farklı yaklaşımla incelenir [1].

• **Her bir karenin bağımsız incelendiği yaklaşım:** Bu yaklaşım klasik fotoğraflardan yüz tanımadan farksız olup, her bir kare birbirinden bağımsız birer resim olarak ele

alınır ve resimdeki kişi veya kişiler tanınmaya çalışılır. Her bir karenin bir gözlem olarak ele alındığı bu yapılarda, tekli gözlemler üzerinde çalışan sınıflandırıcılar (En Yakın K Komşu, Destek Vektör Makineleri vb.) kullanılır.

• **Karelerin Aynı Anda İncelendiği Yaklaşım:** Bu yaklaşımda ise kareler arasındaki ilişki de göz önüne alınarak karelerin, dolayısıyla karelerdeki aynı kişiye ait yüzlerin zamana göre konumları ve değişimleri incelenir. Burada Gizli-Markov Modeli gibi klasik makine öğrenme yöntemlerinin yanında Yayılma Yaklaşımı (Condensation Approach) gibi özgün yöntemler veya yüzün üç boyutlu modellenmesi gibi çalışmalar gerçekleştirilmektedir.

1.1. Sitemin Çalışma Mantığı

Projede yukarıda bahsedilen yaklaşımlardan 1.si kullanılmaktadır. Öncelikle çeşitli duygu durumları belirlenen binlerce resim kümesinden öğrenme işlemi gerçekleştirilir. Öğrenmenin ardından bu resim kümesisi içerisinde bulunan test için ayrılmış olan resimler yardımı ile test işlemi gerçekleştirilir. Bu süreç eğitim sürecidir. Eğitilen programımız kameradan aldığı görüntüyü değerlendirerek daha önce eğitildiği resimlerden hangilerine yüzdesel olarak daha çok benziyor ise tahminde bulunarak bize sonuç döndürmektedir.

Burada kullanılan resim kümesi 30000 den fazla resim BGR gri renk formatında piksellere ayrılmış olarak hazırlanmıştır. Resimler 7 farklı duygu durumuna göre sınıflandırılmıştır. Bu duygu durumları Normal, Korkmuş, Mutlu, Şaşkın, Kızgın, Nefret, Üzgün olarak belirlenmiştir. Resim kümesi içerisinde bulunan bu resimlerin büyük bölümü kullanılarak programımız eğitilir. Kümenin geri kalan resimleri ile ise test yapılır. Yapılan 30 eğitimin sonucunda %70 doğruluk oranında tahminler yapıldığı ortaya çıkmıştır. Eğitim ve test sonucunda elde edilen veriler kaydedilmiştir.

Uygulama çalıştığı an ekranda bir yüz bulmaya çalışmaktadır. Eğer bir yüz tespiti yapılırsa bu yüz bir dikdörtgen içerisine alınarak üzerine hangi duygu durumuna sahip olduğunun tahmini yazılır. Bu süreçte arka planda tespit edilen yüz ile ilgili çeşitli işlemler yapılır. Tespit edilen yüzün BGR renk formatına dönüştürülmesi, 48x48

piksel boyutlarında kesilmesi dönüştürülmesi gibi. Bu işlemlere 3. Bölümde ayrıntılı olarak değinilmiştir.

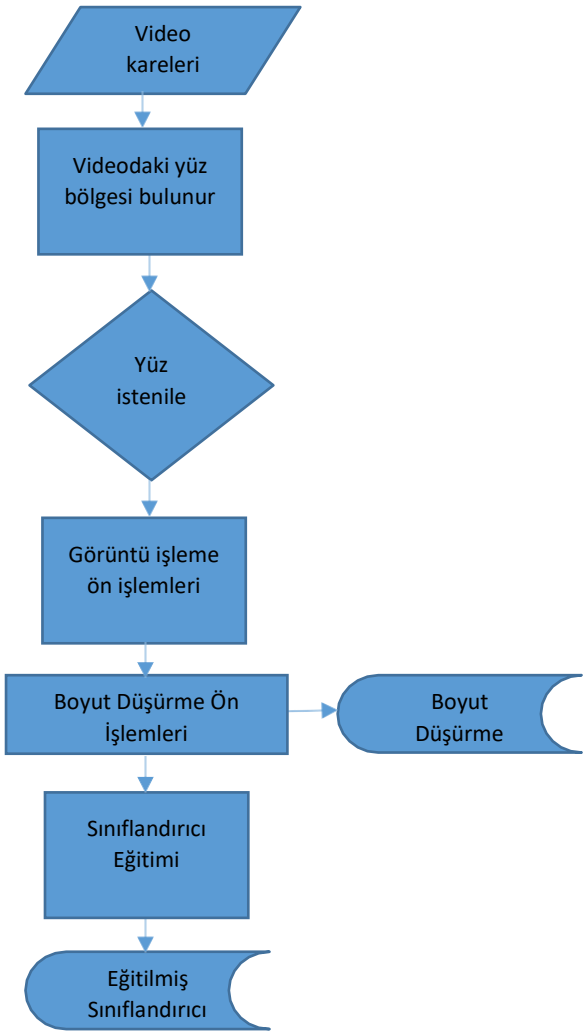
2. PROJE KONUSU HAKKINDA GENEL BİLGİLER

Kurulan sistem, eğitim ve test aşamasında benzer adımları gerçekleştirerek, bir sınıflandırıcı eğitir ve bu sınıflandırıcı ile yeni videoların tanınması gerçekleştirilir. Kullanılan algoritmaların detaylı anlatımları sonraki bölümlere bırakılmak üzere, sırasıyla eğitim ve test aşamalarının çalışma düzenlerine bu bölümde kısaca değinilecektir. [2]

2.1 Eğitim Aşaması

Eğitim aşaması, sistemde daha önceden kime ait oldukları bilinen kişilerin videolarının işlenmesi ve bu videolardan elde edilerek ön işlemlerden geçirilen veriler ile bir sınıflandırıcı eğitilmesi işlemlerini kapsamaktadır. Bu aşamanın işlemleri kısaca şu şekilde özetlenebilir:

1. Tüm eğitim videoları için tüm kareler tek tek işlenir. Her karede varsa yüz bölgesi bulunur ve ayırt edici bir kare ise ön işlemlerden geçirilir.
2. Seçilen ve ön işlemlerden geçirilen yüz resimlerinden oluşturulan veri setinde boyut dönüşümü için gerekli değerler hesaplanır ve dönüşüm uygulanır.
3. Boyut dönüşümü uygulanmış eğitim seti üzerinde bir sınıflandırıcı eğitilir. Aynı işlemler bir akış şemasında ifade edilirse, aşağıdaki gibi bir şema elde edilir:



Şekil 2.1 Eğitim Aşaması

2.2 Sınıflandırma / Test Aşaması

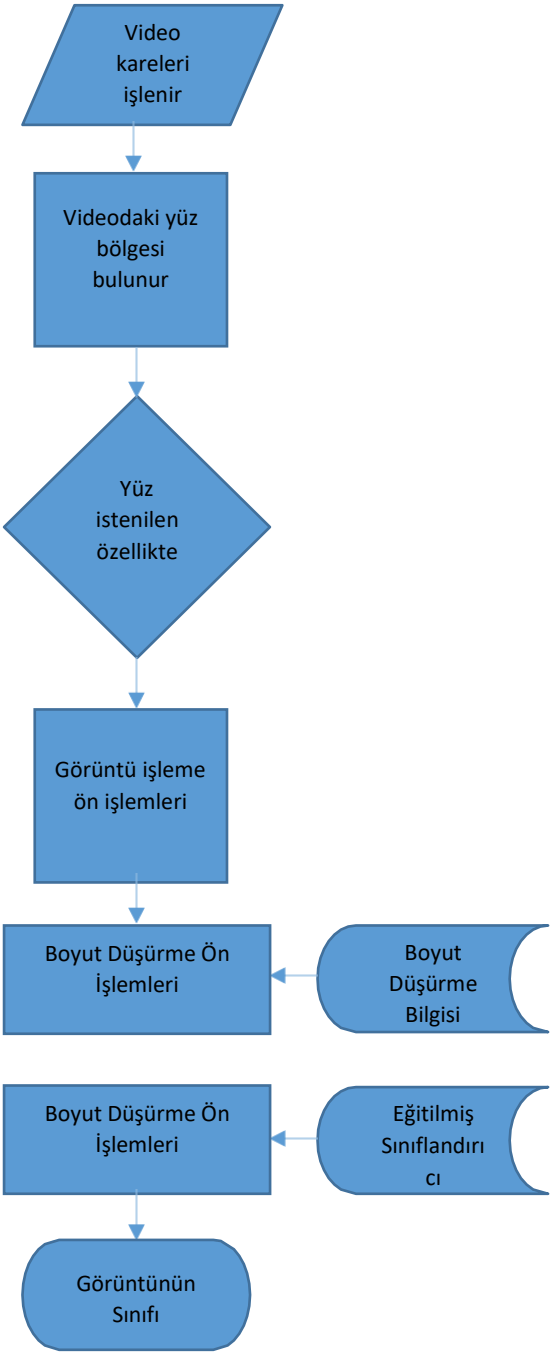
Sınıflandırıcı eğitimi ile tamamlanan eğitim aşaması sonrasında, sistem tarafından bir videonun tanınması ise aşağıdaki şekilde gerçekleştirilir:

1. Videonun her karesini işleyerek varsa ayırt edici yüz kareleri seçilir ve ön işlemler uygulanır.

2. Eğitim aşamasında hesaplanan dönüşüm değerleri, seçilen ve ön işlemlerden geçirilen karelerden oluşturulan veri setine uygulanır.

3. Her kare tek tek sınıflandırılır ve tüm karelerin sınıf bilgileri kullanılarak videonun sınıfı hesaplanır.

Bu aşamanın işlemleri bir akış şemasında ifade edilirse, aşağıdaki gibi bir şema elde edilir:



Şekil 2.2 Sınıflandırma / Test Aşaması

Yapay zeka **alanı, makinelerin deneyimle öğrenebileceği ve insan katılımı olmadan beceriler kazanabileceği** makine **öğrenimini kapsıyor.** Derin öğrenme **ise,** yapay **sinir ağlarının ve insan beyninden ilham alan algoritmaların veriden öğrendiği bir** makine **öğreniminin alt kümesidir**

Artırılmış gerçeklik, gerçekliğin bilg. tarafından değiştirilmesi ve artılırılması. Gerçek dünya + sanal dünya'yı bir araya getirir. Örn snapchat filtreleri.

3. PROJE ÇALIŞMASINDA KULLANILAN MATERYALLER

3.1. OpenCV Kütüphanesi

OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir.

OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme,

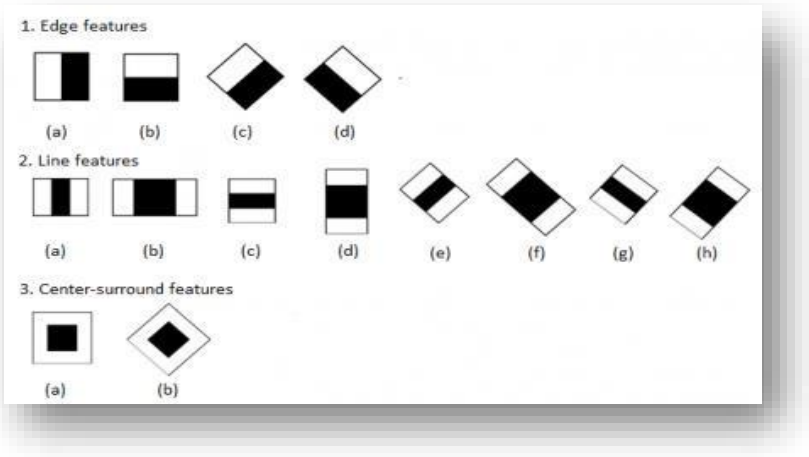
görüntü karşılaştırma, optik karakter tanımlama OCR (Optical Character Recognition) gibi işlemler rahatlıkla yapılabilmektedir [3].

Şekil 3.1. OpenCV

3.2. HaarCascade Sınıflandırıcısı

Opencv kütüphanesi içinde bulabileceğiniz haarcascade sınıflandırıcısı; Paul Viola ve Micheal Jones tarafından, bu soruna çözüm için oluşturulmuştur. Aynı zamanda Viola and Jones object detection framework (Viola ve Jones nesne bulma yapısı) olarak da bilinmektedir. En temel manada belirli bir algoritmaya göre bulunması istenen nesneler önce bilgisayara tanıtılır ve daha sonra ona benzer şekillerin bulunduğu resimler veya video frameleri taranarak o nesne bulunmaya çalışılmaktadır.

Eğitim için içerisinde aranılan nesnenin bulunduğu pozitif resimlere ve içerisinde o nesnenin bulunmadığı negatif resimlere ihtiyaç vardır. Sınıflandırıcı eğitimde, pozitif resimlerdeki nesneleri aşağıdaki gibi belirli büyüklerde ayarlanmış çerçevelerle tarayarak çerçeve içerisinde bulunan siyah bölgedeki piksel değerleri toplamı ile beyaz bölgedeki piksel değerleri toplamlarından karanlık aydınlık değerler kontrol edilerek belirli hedef değerler oluşturulmaktadır.



Şekil 3.2. Haarcascade çerçeveler

Feature denilen bu çerçevelere zayıf sınıflandırıcılar denilmektedir. Çünkü tek başına doğru bir sınıflandırıcı olamazlar. Bir nesnede bu zayıf sınıflandırıcılardan birçoğu olacaktır ve bu zayıf sınıflandırıcıların toplandığı noktada büyük doğruluk oranıyla aranılan nesne var demektir. Sınıflandırıcı, en temel mantığıyla bu şekilde çalışmaktadır.

Çerçeveler aşağıdaki gibi örnek pozitif resimler üzerinde taranır.

Şekil 3.3. Haarcascade çerçevelerin resim üzerinde gösterimi

Yukarıdaki çerçeve için yanakların parlaklık oranının burun bölgesindeki parlaklık oranından daha düşük olması ile burun kısmı seçilebilmektedir.

Şekil 3.4. Haarcascade çerçevelerin resim üzerinde gösterimi

Aynı zamanda göz bölgesinin beyaz ile gösterilen alt bölgeden daha karanlık olması da bu özelliklerden bir tanesidir. Haarcascade sınıflandırıcısında, buna benzer birçok özellikler içinde nesnenin bulunduğu resimler üzerinden geçirilerek değerler oluşturulur. Örneğin yüz taramasında ağız, burun, alın, saç gibi bölgelerde birçok karanlık aydınlık özellikleri oluşturulacaktır. Bunların her birinden hedef değerler oluşturulmaktadır. Ve bu işlem çerçeve büyüklükleri değiştirilerek diğer aşamalarda tekrar edilmektedir.

Bu çerçeveler (zayıf sınıflandırıcılar) her resim boyutu için düşünüldüğünde yüz binlerce çekirdek oluşacaktır. Negatif resimler üzerinde tarama yapılarak içinde nesne bulunmadığı için kullanılmayacak olan çerçevelerin büyük çoğunluğu elenecektir. Pozitif resimlerde nesne seçilerek nesnedeki kullanılacak çerçeveler belirlenecektir. Bunun için eğitim sırasında pozitif resimlerde nesnenin milimetrik seçimine dikkat edilmelidir. Pozitif ve negatif resim örneklerinin çok olması istenilen nesnede daha iyi sonuçlar almak için önemlidir.

Bu işlemlerin hem eğitimde hem de nesnelerin bulunmasında bilgisayarı çok yoracağı ve işlemlerin uzun süreceği düşünülebilir. Reel time görüntü işlemede hız çok önemlidir. Haarcascade sınıflandırıcısında öncelikle resimlerin integralleri alınır. Böylelikle piksel değerlerinin tek tek toplamları hesaplanmak yerine integralle hesaplanmış olmaktadır. Böylelikle bilgisayardan büyük bir işlem gücü kaldırılacaktır. Ayrıca nesnelerin bulunması aşamasında da her bir büyüklükteki çerçeve tarafından tekrar tekrar taranmak yerine sadece önceki aşamalarla eşleşme olan kısımlar taranarak bir işlem yükü de oradan azaltılmaktadır. Bu açılardan ve yaptığı iş açısından oldukça hızlı oldukları söylense de uygulama aşamasında reel time çalışıldığında alınan

görüntüyü belirli oranda yavaşlattığı görülmektedir. Bu hız, sınıflandırıcının eğitilme şekli, örnek sayısı gibi durumlara göre değişmektedir.

İstenilen nesnelerin bulunabilmesi için sınıflandırıcının referans alacağı min hit rate ve max false alarm rate değerleri vardır. Sınıflandırıcı eğitimde her seferinde belirli eğitim algoritması ile bu değerlere ulaşmaya çalışır. Minhitrate minimum isabet oranını ve max false alarm rate ise hangi hata oranındaki nesnelerin gösterileceğini ifade eder.

Eğitim yapıldıktan sonra .xml uzantılı bir dosya oluşturulacak ve bu dosya ile opencv kütüphanesi kullanılarak istenilen nesne bulunabilecektir [4].

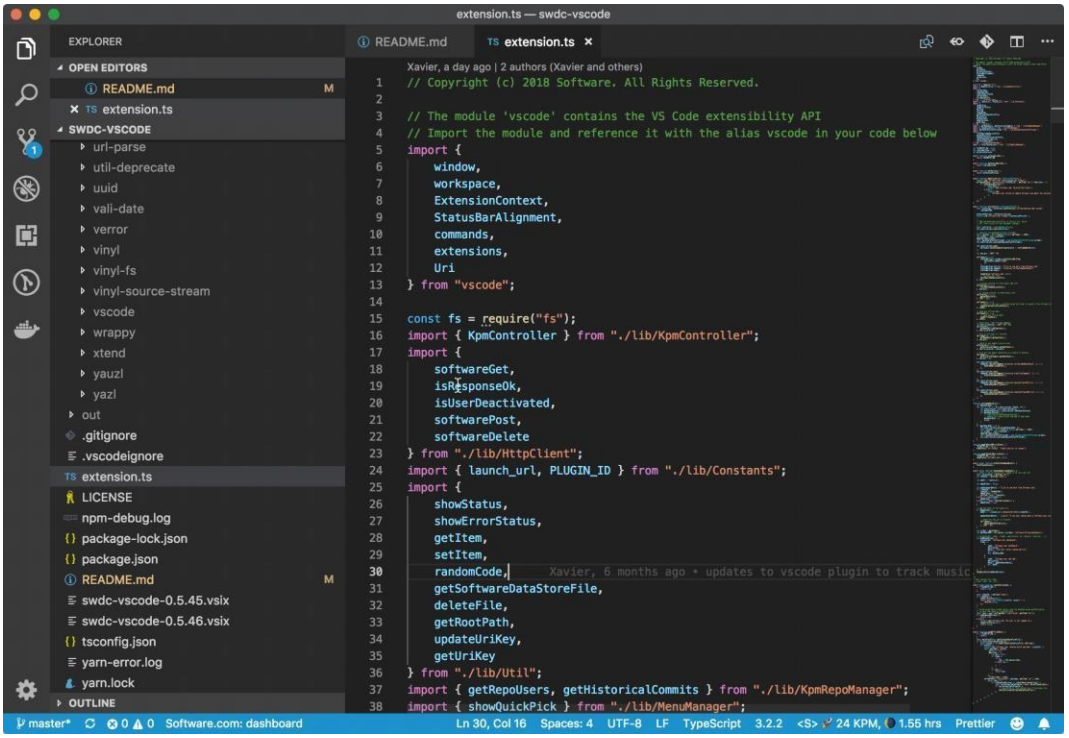
3.3 Visual Studio Code

Günümüzde programlama dilleri bağımsız bir alana taşınmaya başlanmıştır. Bir programlama dili ile geliştirme yapmak için her türlü işletim sistemi kullanılabilir olmalıdır. Son bir kaç yılda Microsoft bu değişime ayak uydurarak açık kaynak ürünler ve platform bağımsız teknolojiler üretmeye başlamıştır. Microsoft .net ile artık Mac OS ve Linux platformlarında geliştirme yapmak mümkün hale gelmiştir.

Visual Studio Code hızlı, hafif olmanın yanında Microsoft, Linux ve Mac işletim sistemlerinde çalışabilen bir araçtır. Kurulum yapıldığında basit bir metin editörü görüntüsü ile karşımıza çıkan ürün, eklentiler sayesinde Node JS, Ruby, Python, C/C++, C#, Javascript gibi bir çok programlama dilini desteklemektedir. Yani klasik Visual Studio ürünleri gibi her şeyi bünyesinde barındırmak yerine çekirdek bir yapı ile başlatılıp lazım olan parçaların eklentiler halinde ürüne eklenmesi sağlanarak mantıklı bir hareket yapılmıştır. Eklentilerin indirilebildiği bir market ortamı oluşturulmuştur.

Visual Studio Code, hata ayıklama özellikleriyle, gelişmiş web ve bulut uygulamaları üstünde kodları düzenlemeye, yeniden tanımlamaya ve optimize etmeye yarar. Visual Studio Code tamamen ücretsiz olup, dilediğiniz gibi kullanabilir, kodlarını inceleyebilir ve kendi ihtiyaçlarınıza göre değişim yapabilirsiniz. Uygulama, çoklu platform esteğine sahip olduğu için Linux, Mac OS

X ve Windows üzerinde çalışır ve programcılar için yaklaşık 30 programlama dili desteği sunmaktadır [5].

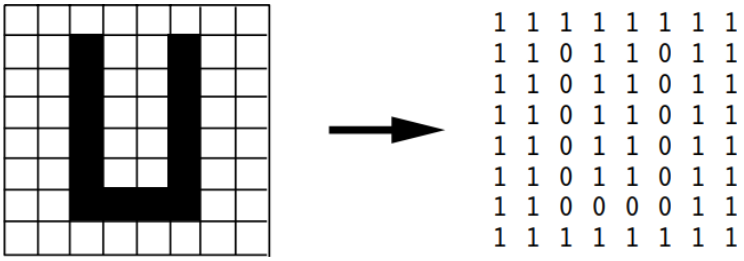


Şekil 3.5. Visual Studio Code Ekranı

3.4 CNN (Konvolüsyonel Sinir Ağları)

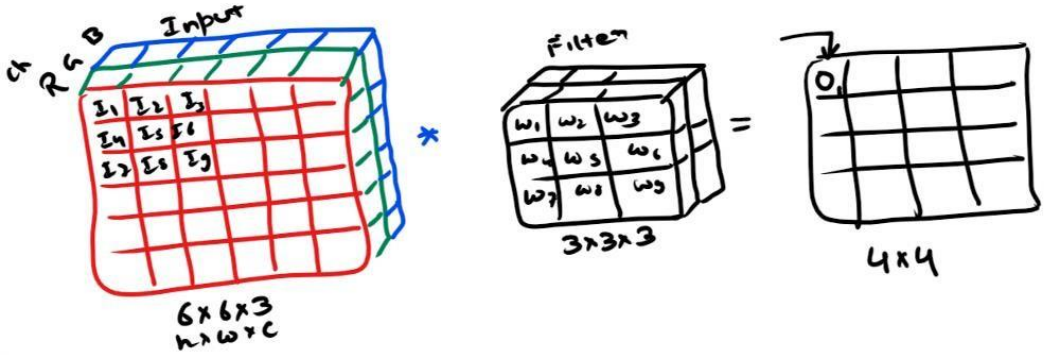
Input, CNN için geçerli input dosya biçimi görüntüdür. Bu görüntüler (resimler) sadece sayılardan oluşmaktadır. Siyah-Beyaz resim dediğimiz sadece 2 çeşit renkten oluşur ki bunlar siyah ve beyazdır. Sayısal değer olarak (yani bilgisayarın gözünde) bunlar 0 ve 1 değerleridir. Siyah Beyaz resimler 2 Boyutlu matrislerle ifade edilirler.

0 siyahı temsil ederken 1 beyazı temsil eder. [6]



Şekil 3.1. Görselin Matrislerle Gösterimi

Konvolüsyon işleminde gerçek sihir burada gerçekleşir. Konvolüsyon, iki bilgi kümesini birleştiren bir işlemdir. Burada konvolüsyon, bir özellik haritası (feature map) üretmek için convolution filter kullanarak input matrisine uygulanır. Input matrisinin renkli resimlerde 3 boyutlu olduğunu yukarıda açıklanmıştır.

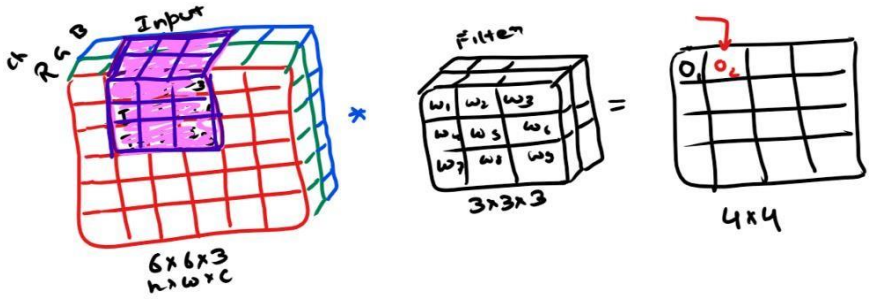


Şekil 3.2. Konvolüasyon İşlemi

Yukarıdaki resimde sol tarafta 6x6x3 boyutunda küçük bir resim bizim veri setimizdeki eğitmek için kullandığımız resimlerdir**.** Konvolüsyonel katmanda bütün değerler sırasıyla çarpılıp toplanır ve sonuçta bir değer elde edilir.

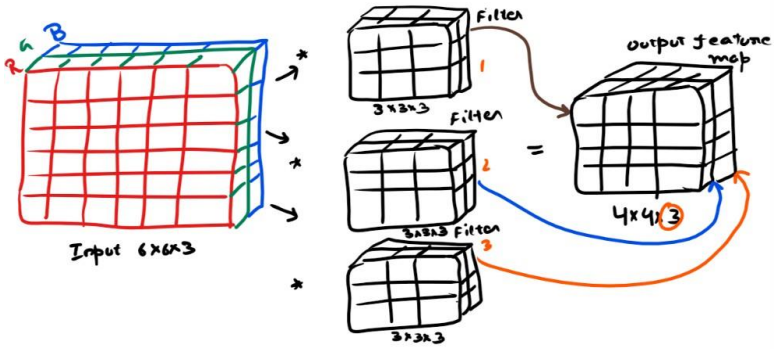
$$(I_1 \times w_1) + (I_2 \times w_2) + \ldots + (I_9 \times w_9) = O_1 \qquad \text{Eşitlik (3.1.)}$$

Stride, filtrenin input görüntüsü üzerinde ne kadar (kaç birim) hareket ettireceğimizi belirler. Adım 1'e eşitse, filtre bir kerede bir piksel / birim kaydırılır.



Şekil 3.3 Konvolüasyon İşlemi

Filtre sayısının çokluğu geriye output olarak bir dizi şeklinde dönecektir. Yani; nasıl ki renkli görüntüde 3 derinlik (RGB) var ise çıktıda da derinliğin sayısı filtrenin sayısına eşit olur.



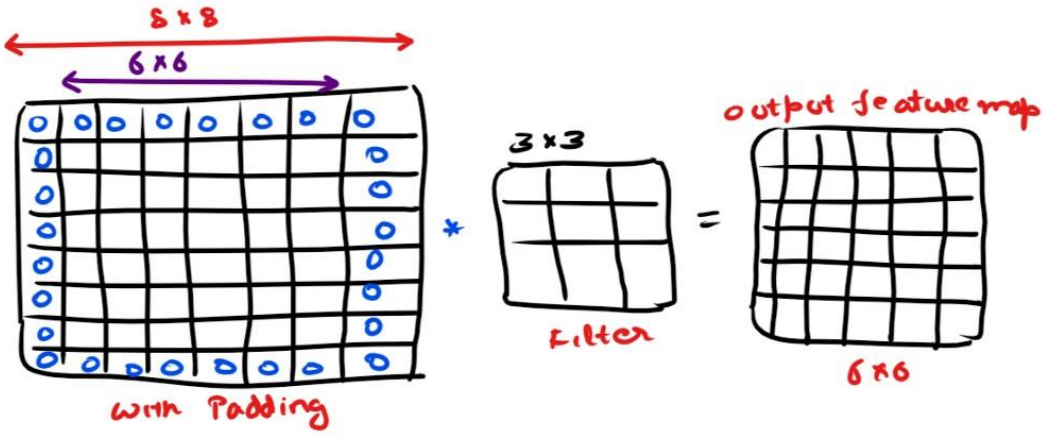
Şekil 3.4 Konvolüasyon İşlemi

4 x 4 olmasının sebebi ise stride'ın 1 olmasıdır. 3 ise 3 filtre olduğu içindir. Yani output 4x4x3 tür. Kısaca konvolüsyon katmanında filtreler ile resim konvolüsyon işlemine tabii tutulmaktadır. Filtre sayısı kadar feature map oluşmaktadır.

Non-linearity/ReLu kendi kendine özgü bir katman değildir, sadece konvolüsyon katmanının bir parçasıdır. Yani burada anlatılanlar convolutional layer'ın devamı niteliğindedir. Konvolüsyon işleminin çıktısına ReLu aktivasyon fonksiyonunu uygulanmaktadır. Son feature map'teki değerler aslında toplamlar değil, üzerlerinde uygulanan Relu işlemidir.

Yani şöyle ki; konvolüsyon işlemini hesapladıktan sonra bulunan değer O1 olsun. Bu, gidip doğruca output olarak O1 yazılmamaktadır. Yukarıdaki örnekten (I_1 x w_1) + (I_2 x w_2) + … + (I_9 x w_9) = O_1 ifadesi tam olarak O_1'e eşit değil. Burada O_1 değeri üzerine ReLu uygulanmakta ve O_1 değeri O_1' değeri olmaktadır. Diyelim ki O_1 değeri hesaplamalara göre 5 olsun. Output değeri olarak 5 değil ReLu uygulanıp (örnek) 4.20 olmaktadır.

Padding demek 0'larla matrisin genişliğini veya yüksekliğini artırmak demektir. Stride yaparken 6×6'lık input resmi konvolüsyon yaptıktan sonra output olarak 4×4'lük boyutta olmaktaydı. Matriste bir küçülme (kayıp) olmaktaydı. Bunu engellemek için 6×6'lık matrisin hacmini 0'larla artırıyoruz ki (8×8 yaparak) küçülme olduğunda 8×8'lik matris 6×6'lık matris olsun. Böylece orijinal boyut korunmaktadır.



Şekil 3.5 Padding İşlemi

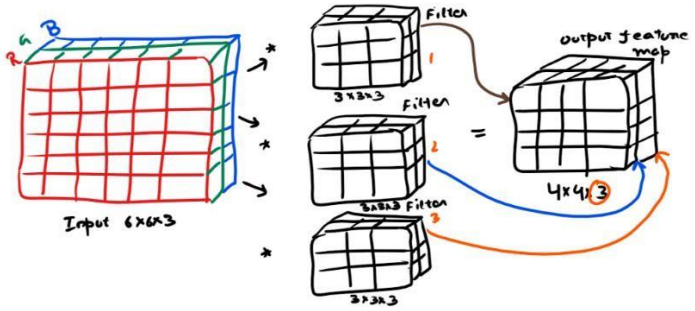
Pooling; öğrenme süresini kısaltmakta ve overfitting olmasını engellemektedir. (Burada yapılan iş ne?) Convolutional Layer'dan çıkan resimlerin boyutunu küçültmektedir. Burada öğrenciler kalın yazılan **"a"** harfi ile ince yazılan "a" harfinin aynı olduğu bilinmektedir. Sonuçta ikisi de aynı anlamı vurgulamaktadır. Burada kalın **"a"** harfini örnekleyerek ince "a" harfi ile örnekleme (sampling) denir.

Max, Min, Mean Pooling; Convolutional Layer'da üretilen feature map'teki değerlerin üzerinde bir mxm'lik bölgeyi ele almakta ve içinden maksimum değeri seçmektedir. Min Pooling'de bu değerin en küçük olanını Mean Pooling'de ise mxm'lik bölge içerisindeki bütün değerlerin ortalamasını almaktadır. Böylece tek bir değere indirgenmiş olmaktadır.

Flattening Layer katmanı aslında Fully Connected katmanı ile birlikte ele alınır fakat bu çalışmada ayrı olarak ele almak tercih edilmiştir. Şimdiye kadar görüntümüz hep matris şeklindeydi. Bu katmanda basitçe tek boyutlu olan vektöre dönüştürülmektedir. Vektöre dönüştürme işlemini flattening layer otomatik olarak yapmaktadır. Boyut değiştirme aşaması olarak da düşünülebilir. Vektöre dönüştürmesindeki amaç ANN'de kolayca eğitebilmek içindir.

Fully Connected Layers, CNN dediğimiz mimarinin ANN kısmıdır ve son katmandır. Burada gerçekleşenlerin ANN adlı yazımdaki anlattığım olaylardan hiçbir farkı yoktur. Bu katman başlı başına bir okyanustur ve CNN'den önce ANN öğrenilmesi muhakkak tavsiye edilmektedir.

Bu aşamalardan geçildikten sonra artık karar verme aşamasına ulaşılmıştır. Input'a göre kedi, köpek vs gibi output değerleri yapılabilir.



Şekil 3.6 Output

4 x 4 olmasının sebebi ise stride'ın 1 olmasıdır. 3 ise 3 filtre olduğu içindir. Yani output 4x4x3 tür. Kısaca konvolüsyon katmanında filtreler ile resim konvolüsyon işlemine tabii tutulmaktadır. Filtre sayısı kadar feature map oluşmaktadır.

3.5. FER2013 Veri Kümesi

Fer2013, Pierre-Luc Carrier ve Aaron Courville tarafından devam etmekte olan bir proje için oluşturulan, daha sonra ICML 2013'ten kısa bir süre önce halka açık bir Kaggle yarışması için paylaşılan açık kaynaklı bir veri kümesidir. Bu veri kümesi, 35.887 gri tonlamalı, 48x48 boyutlu yüz görüntülerinden oluşur. 7 duygu durumu etiketlenmiştir. [7]

Veri kümesindeki duygu etiketleri:

0: -4593 görüntüler- *Kızgın*

1: -547 görüntüler- *Nefret*

2: -5121 görüntüler- *Korku*

3: -8989 görüntüler- *Mutlu*

4: -6077 görüntüler- *Üzgün*

5: -4002 görüntüler- *Şaşkın*

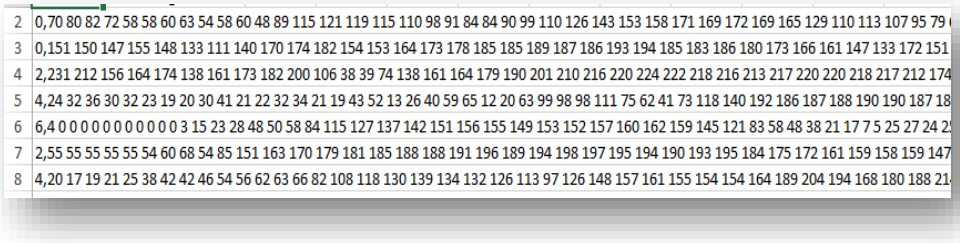
6: -6198 resimler- *Nötr*

Yarışma sırasında 28.709 görüntü eğitim, 3.589 görüntü ise açık test seti olarak katılımcılar ile paylaşılmış ve kalan 3.589 görüntü ise yarışmanın galibini bulmak için özel test seti olarak tutulmuştur. Veri seti, yarışma tamamlandıktan sonra herkesin erişebileceği şekilde ayarlanmıştır.

Şekil 3.6. fer2013 Örnek Resimler

Paylaşılan veri seti içerisinde herbir satırda resimlerin piksellere dönüştürülmüş BGR renk kodları bulunmaktadır. Her satırın ilk sütunu o resme etiketlenmiş olan duygu

durumunu belirtmektedir. Şekil 3.7 de fer2013 isimli veri kümesinin içeriği görüntülenmektedir.



Şekil 3.7 fer2013 İçeriği

Fer2013 veri kümesi ile 30 eğitim gerçekleştirilmiştir. Burada eğitim sayısı ve veri kümesindeki veri miktarının arttırılmasıyla tahmin oranınıda arttığı görülmüştür. Örneklem olarak 1.,10.,18.,25. ve 30. eğitimlerin sonuçları görülmektedir. 1. eğitimdeki doğruluk oranı %30 civarında iken 30. Eğitim sonunda %70 civarında doğruluk elde edilmiştir.

Train on 28709 samples, validate on 3589 samples

Epoch 1/30

28709/28709 [==============================] - 460s 16ms/step - loss: 1.7062 - **accuracy: 0.3050** - val_loss: 1.5514 - val_accuracy: 0.3870

Epoch 10/30

28709/28709 [==============================] - 466s 16ms/step - loss: 1.1379 - **accuracy: 0.5684** - val_loss: 1.1861 - val_accuracy: 0.5556

Epoch 18/30

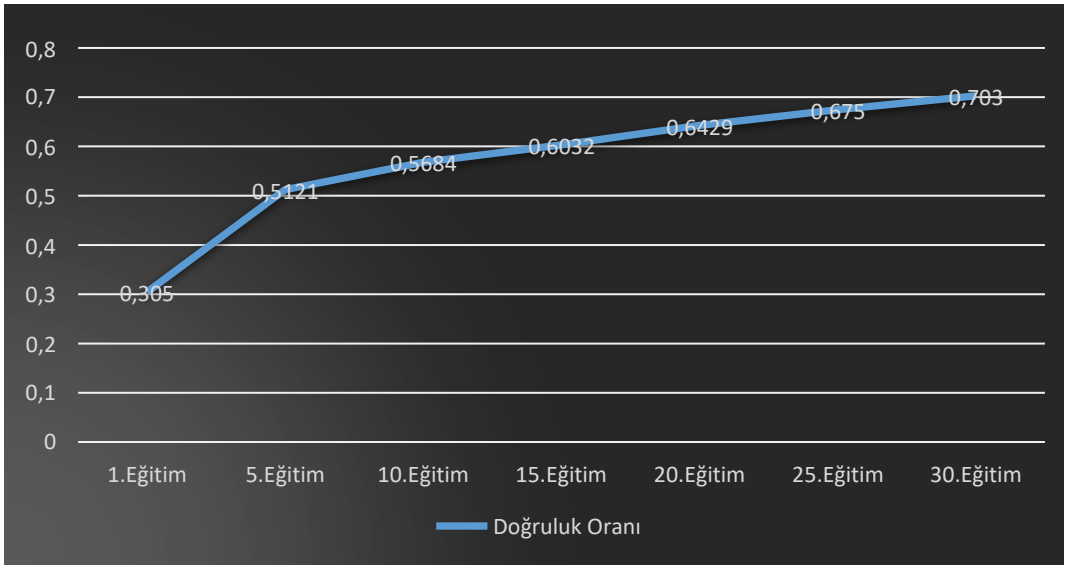
28709/28709 [==============================] - 506s 18ms/step - loss: 0.9753 - **accuracy: 0.6279** - val_loss: 1.1715 - val_accuracy: 0.5762

Epoch 25/30

28709/28709 [==============================] - 506s 18ms/step - loss: 0.8560 - **accuracy: 0.6750** - val_loss: 1.2149 - val_accuracy: 0.5706

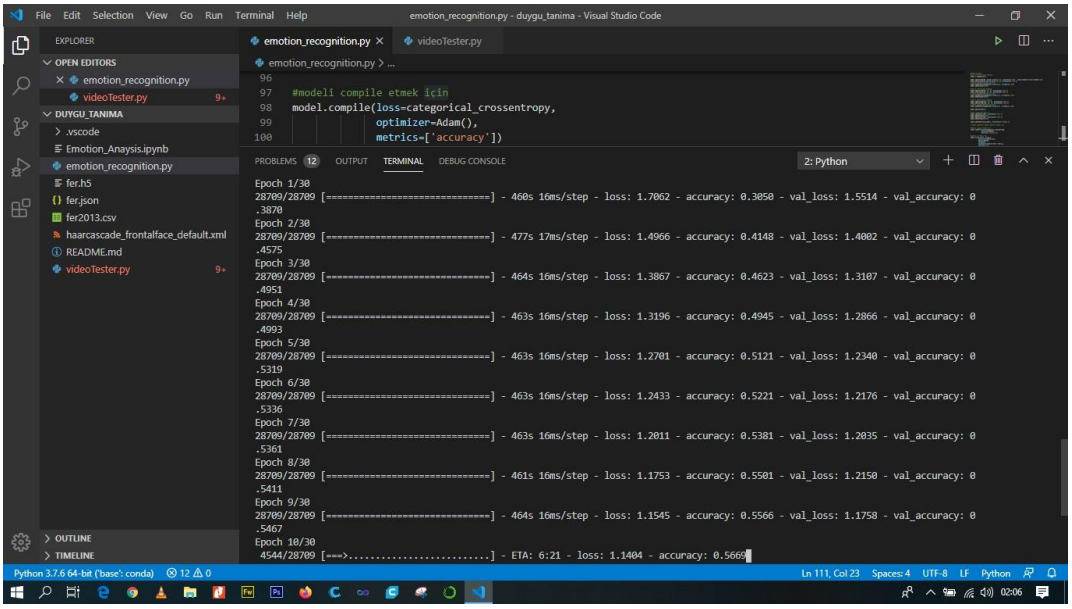
Epoch 30/30

28709/28709 [==============================] - 510s 18ms/step - loss: 0.7902 - **accuracy: 0.7030** - val_loss: 1.2442 - val_accuracy: 0.5756



Grafik 3.1 Eğitim Doğruluk Oranı

Eğitim kodları çalıştırıldığında 30 eğitim için yaklaşık iki buçuk saat gibi bir sonunda eğitim tamamlanmaktadır. (Bilgisayar donanım özelliklerine göre değişiklik gösterebilir.) Her eğitimin sonunda doğru tahmin oranı artmaktadır. Bu süreç tepe değere geldiğinde doğruluk oranında kayda değer bir değişiklik göstermemektedir. Doğruluk oranının artması noktasında daha farklı ve çok sayıda resim veri tabanına ihtiyaç duyulmaktadır.



Şekil 3.8 Eğitim ve Test Ekranı

3.6. Eğitim ve Test Program Kodları

Aşağıda verilen eğitim için gerekli kod kümesi incelendiğinde ilk olarak gerekli kütüphanelerin tanımlandığını görülmektedir. Ardından kaynak dosyasının olan fer2013 dosyası okutulmaktadır. Resimlerin tanımlanması işlemlerinin ardından eğitim sayısı (epochs) belirlenmektedir. CNN tasarımında e defa konvolüasyon işlemine tabi tutulup nöral ağı oluşturulmaktadır. Daha sonra modeli compile ederek eğitim işlemi gerçekleştirilmektedir. Eğitim sonunda eğitilen model kaydedilmektedir.

```
import sys, os
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization,Avera
gePooling2D
from keras.losses import categorical_crossentropy
from keras.optimizers import Adam
from keras.regularizers import l2
from keras.utils import np_utils
df=pd.read_csv("C:/duygu_tanima/fer2013.csv")
```

```
X_train,train_y,X_test,test_y=[],[],[],[]
for index, row in df.iterrows():
    val=row['pixels'].split(" ")
    try:
        if 'Training' in row['Usage']:
           X_train.append(np.array(val,'float32'))
           train_y.append(row['emotion'])
        elif 'PublicTest' in row['Usage']:
           X_test.append(np.array(val,'float32'))
           test_y.append(row['emotion'])
    except:
        print(f"error occured at index :{index} and row:{row}")
num_features = 64
num_labels = 7
batch_size = 64
epochs = 30
width, height = 48, 48
X_train = np.array(X_train,'float32')
train_y = np.array(train_y,'float32')
X_test = np.array(X_test,'float32')
test_y = np.array(test_y,'float32')
train_y=np_utils.to_categorical(train_y, num_classes=num_labels)
test_y=np_utils.to_categorical(test_y, num_classes=num_labels)
X_train -= np.mean(X_train, axis=0)
X_train /= np.std(X_train, axis=0)
X_test -= np.mean(X_test, axis=0)
X_test /= np.std(X_test, axis=0)
X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)
X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)
model = Sequential()
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape
=(X_train.shape[1:])))
model.add(Conv2D(64,kernel_size= (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_labels, activation='softmax'))
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(),
              metrics=['accuracy'])
model.fit(X_train, train_y,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_test, test_y),
          shuffle=True)
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer.h5")
```

3.7. Video Yakalama Program Kodları

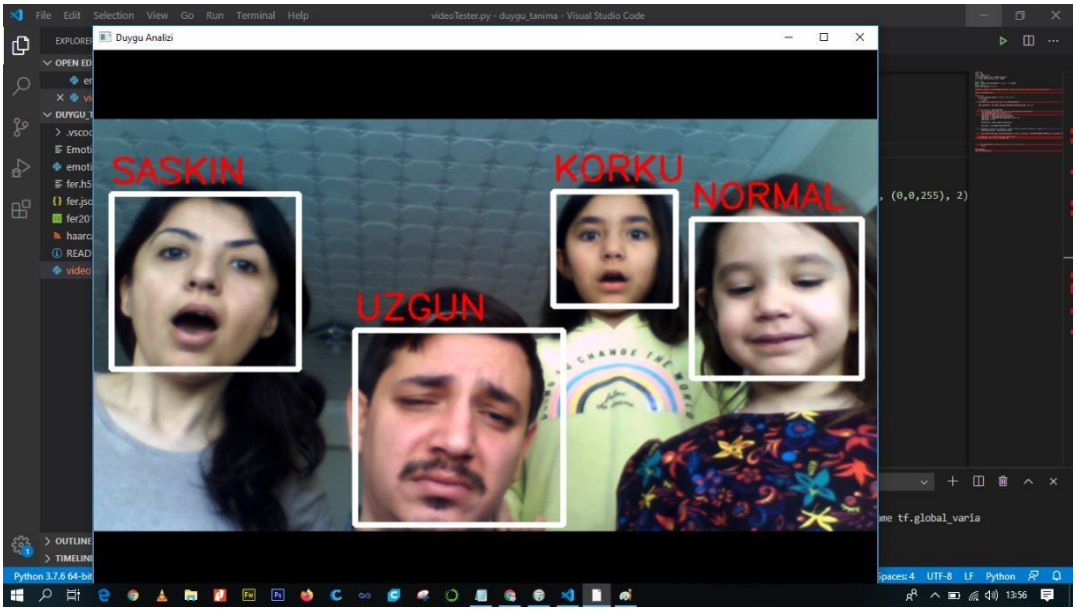
```
# Dosya ve klasör yapılarıyla çalışmak için gerekli  kütüphane
import os
# Yüz tanıma resim alma işlemleri için gerekli kütüphane
import cv2
# Matris ve sayı dizileri için gerekli kütüphane
import numpy as np
# Derin öğrenme için kullanılan kütüphane
from  keras.models  import  model_from_json
```

```
from keras.preprocessing import image
# Modeli Yükle
model = model_from_json(open("fer.json", "r").read())
# Ağırlıkları Yükle
model.load_weights("fer.h5")
# Belirtilen yoldaki HaarCascade xml dosyasını aç ve değişkene ata
face_haar_cascade = cv2.CascadeClassifier("haarcascade_frontalface_defa
ult.xml")
# Videoyu yakala ve değişkene ata
cap=cv2.VideoCapture(0)
# Döngü kur
while True:
    # Videodan resmi yakala
    ret,test_img=cap.read()
    if not ret:
        continue
    # Resmi BGR renk formatına dönüştür ve değişkene ata
    gray_img= cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)
    # Haar Cascade xml dosyasında eşleşme bul ve değişkene ata
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32,
 5)
    for (x,y,w,h) in faces_detected:
        # Dikdörtgen oluştur (Belirlenen renk ve kalınlıkta)
        cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,255,255),thickness=
3)
        # Yüzü kes
        roi_gray=gray_img[y:y+w,x:x+h]
        # Yüzü tekrar boyutlandır
        roi_gray=cv2.resize(roi_gray,(48,48))
        # Pixeller üzerinde çalış ve tahmin oluştur
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis = 0)
        img_pixels /= 255
        predictions = model.predict(img_pixels)
        max_index = np.argmax(predictions[0])
        # İfadeleri tanımla
```

```
        emotions = ("KIZGIN ", "NEFRET", "KORKMUS", "MUTLU", "UZGUN", "
SASKIN", "NORMAL")
        # ifadeyi tahmin edilen ifade değişkenine ata
        predicted_emotion = emotions[max_index]
        # Belirtilen özelliklerdeki karakterler ile belirtilen konuma t
ahmin edilen ifadeyi yaz
        cv2.putText(test_img, predicted_emotion, (int(x), int(y-
10)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
    # Test resmini tekrar boyutlandır ve değişkene ata
    resized_img = cv2.resize(test_img, (1000, 600))
    # Dönüştürülen resmi başlıkla ekranda göster
    cv2.imshow("Analiz Edilen Ifade",resized_img)
    # Çıkmak için q tuşuna basın
    if cv2.waitKey(10) == ord('q'):
        break
    # Döngüden çık ve pencereleri kapat
cap.release()
cv2.destroyAllWindows()
```

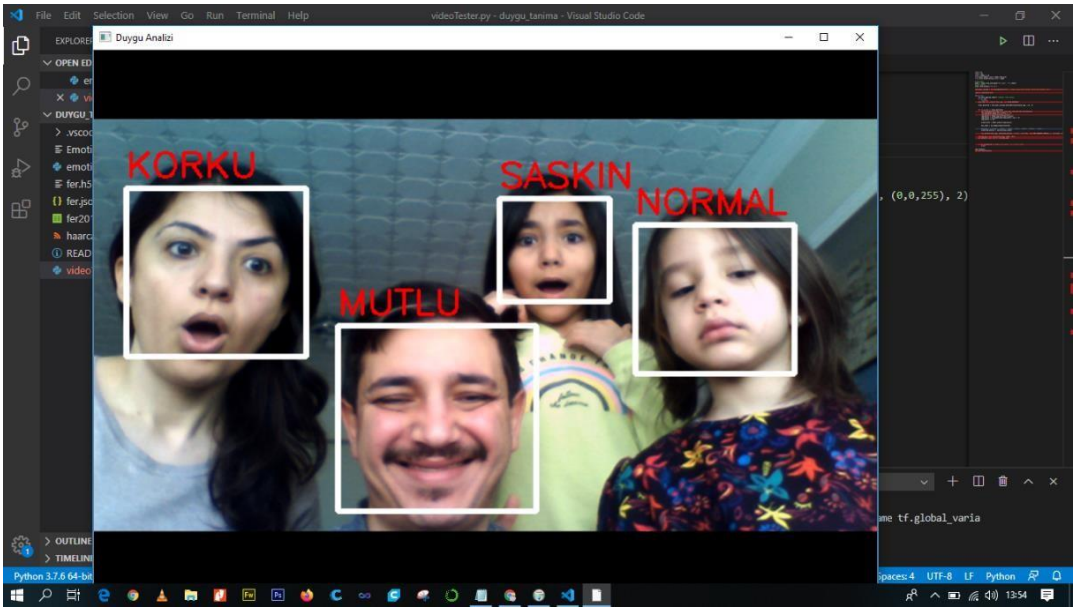
Program kodları incelendiğinde öncelikle dosya ve klasör yapılarıyla çalışmamızı sağlayacak os kütüphanesi, kamera ve resim alma işlemleri için cv2 kütüphanesi, matris ve sayı dizileri ile işlemler yapabilmek için gerekli numpy kütüphanesi ve derin öğrenme için gerekli keres kütüphaneleri import edilmektedir.. Eğitim ve test sonrası oluşturulan modelin (fer.json) ve ağırlıkların (fer.h5) olduğu dosya okutulmaktadır. Ardından sınıflandırma işlemi için gerekli dosya (haarcascade_frontalface_default.xml) okutulup değişkene atanmaktadır. Kamerayı çalıştırıp görüntüden yüz tespit ettirilmektedir. Tespit edilen yüz resmini gri renge dönüştürüp sınıflandırıcıdan eşleşme buldurulmaktadır. Tespit edilen yüz belirlenen renk ve boyutlarda bir çerçeve içerisine alınmaktadır. Yüz kesilerek yeniden boyutlandırılmaktadır. Pixeller üzerinde çalışma yapılarak yüz tanınmaya çalışılmaktadır. Ardından ifadeler tanımlanarak tahmin edilen ifade değişkeni tanımlanmaktadır. Tahmin edilen ifade belirlenen renk boyut ve konumda ekrana

yazılmaktadır. Programdan çıkmak için “q” harfi kullanılmaktadır. “q” harfine basıldığında döngü sonlandırılarak tüm pencereler kapatılmaktadır.



Şekil 3.9 Programın Ekran Görüntüsü

Programın ekran görüntüsüne bakıldığında dört kişinin yüzü tespit edilmiş ve duygu durumları çerçeve üzerine yazılmıştır. Eğitim sonrasında öğrenilmiş olan yüz ifadeleri ile eşleşen tahminler sonucu görüntülerdeki şaşkın, üzgün, korkmuş ve normal ifadeler tespit edilmiştir. Yine aşağıdaki Şekil 3.10 da görüldüğü gibi aynı kişilerin farklı yüz ifadeleri de başarı ile tahmin edilmiştir. Şekil 3.9 da üzgün yüz ifademi tahmin etmiş Şekil 3.10 da ise mutlu yüz ifademi doğru bir şekilde tahmin etmiştir. Yüz yakalama sürecinde daha önce bahsettiğimiz birçok etken mevcuttur. Yapılan denemeler sonucunda en büyük faktörün ışık olduğu kanısına varılmıştır. Kameraya gelen ışık miktarı düştükçe tahmin oranının da doğru orantılı olarak düştüğü, yeteri kadar ışık olmayan ortamlarda hatalı tahmin oranının arttığı tespit edilmiştir. Ayrıca eğitimlerde kullanılan örnek resim sayılarının da artmanı tahminlerde doğruluk oranını önemli ölçüde arttırmaktadır. Örneğin nefret yüz ifadeli resimlerin sayısı az olduğundan bu ifadeyi tahmin etme olasılığı oldukça düşmektedir. Genellikle nefret yüz ifadesine yakın olan kızgın veya üzgün yüz ifadelerine tahmin eğilimi artmaktadır. Eğitimde kullanılan resimlerin çok olması doğru tahmin olasılığını büyük ölçüde etkilemektedir.



Şekil 3.10 Programın Ekran Görüntüsü

4. SONUÇLAR VE ÖNERİLER

Bu proje çalışmasının sonucunda günümüzde kullanımı giderek artan yüz tanıma uygulamalarının temel mantığı hakkında bilgi sahibi olunmuştur. Projeyi çeşitli ortamlarda kullanma imkanı bulunmaktadır. İş hayatında, trafikte ve eğitim ortamları gibi toplumda insan psikolojisini tespit etmek yamaçlı yaptığım bu çalışma daha da geliştirilebilmektedir. Örnek olarak hassasiyet arttırılarak bir iş yerinde çalışan personellerin duygu durumları tespit edilerek duygu durumları hakkında bilgi toplanabilmektedir. Aynı şekilde bir okuldan öğrencilerin duygu durumları hakkında bilgi toplanarak gerekli psikolojik destek verilebilmektedir. Okulun girişine adapte edilecek yüz tanıma cihazına benzer bir kamera sistemi ile öğrencilerin okul girişinde yüzleri tespit edebilmektedir. Olumsuz duygu durumuna sahip öğrencilerin gerekli psikolojik destek alması sağlanabilmektedir. Aynı mantık ile bir işyeri çalışanların motivasyonunu arttırma konusunda önemli rol oynayabilmektedir.

Bu projenin geliştirilmesi noktasında en önemli ekten olan eğitimde kullanılacak olan havuzun çok ayrıntılı ve geniş olmasıdır. Projede kullanılan 30 bin resim yerine 100 bin resim olması tahmin oranını pozitif yönde etkileyecektir. Aynı zamanda resimlerdeki ifade çeşitliliğide orantılı olması uygun olacaktır. Büyük çoğunluğun aynı ifade olması tahminlerde bu ifadeyi bulma ihtimalinin artmasına sebep olacaktır. Aynı zamanda cinsiyet, fiziksel değişiklik (ten rengi, iskelet yapısı, sakal, bıyık v.b.), kullanılan aksesuar (gözlük, hızma v.b.) resimlerin seçiminde göz önünde bulundurulmalıdır.

Havuzun geniş olması tek başına yeterli olmayacaktır. Bu havuzun etkin bir şekilde kullanılabilmesi için yeteri kadar eğitim yapılması gerekecektir. Eğitim sayısının yine fazla olması tahmin oranını olumlu yönde etkileyecektir. Grafik 3.1 de de görüldüğü üzere eğitim sayısı ile tahmin oranının bir noktaya kadar doğru orantılı olduğu görülmektedir.

Yüksek çözünürlüklü ve düşük ortam aydınlatmasında bile net görüntüler elde edebilen bir kamera kullanılması projenin doğru sonuçlar vermesinde etkili olacaktır. Aynı zamanda ortam aydınlatmasınında iyi olması önemli bir etkendir.

KAYNAKLAR

1. Graham, R., McCabe, H. and Sheridan, S., "Pathfinding in computer games", ***Institute of Technology Blanchardstown Journal***, Dublin, Ireland, 8: 56-80 (2003).

2. Kamphuis, A. and Overmars, M. H., "High quality navigation in computer games", ***Science of Computer Programming***, Utrecht, The Netherlands, 67: 91–104 (2007).

3. Bakkes, C. J., Spronck, H. M. and van Lankveld, G., "Player behavioural modelling for video games", ***Entertainment Computing***, Tilburg, The Netherlands, 3 (3): 71-79 (2012).

4. Wang, J. Y., "An effective method of pathfinding in a car racing game", ***The 2nd International Conference on Computer and Automation Engineering (ICCAE)***, Tauyuan, Taiwan, 3: 13-18 (2010).

5. Waveren, J. M. P., "The quake III arena bot", M. Sc. Thesis, ***Delft University of Technology***, Delft, Netherlands, 10-15 (2001).

6. İnternet: Grenier, M., "Pathfinding 103", **http://mgrenier.me/2011/04/pathfinding-103/** (2012).

ÖZGEÇMİŞ

Adı SOYADI 1976'da İzmir'de doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı; İzmir Meslek Lisesi'nden mezun olduktan sonra 1999 yılında Gazi Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne girdi; 2003'de mezun olduktan sonra Kastamonu Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak göreve başladı.

ADRES BİLGİLERİ

Adres: İzmir Meslek Lisesi.

Anamur Caddesi, No. 12, D: 7A

Merkez / İZMİR

Tel: (232) 234 4545

Faks: (232) 234 4546

E-posta: .....@gmail.com