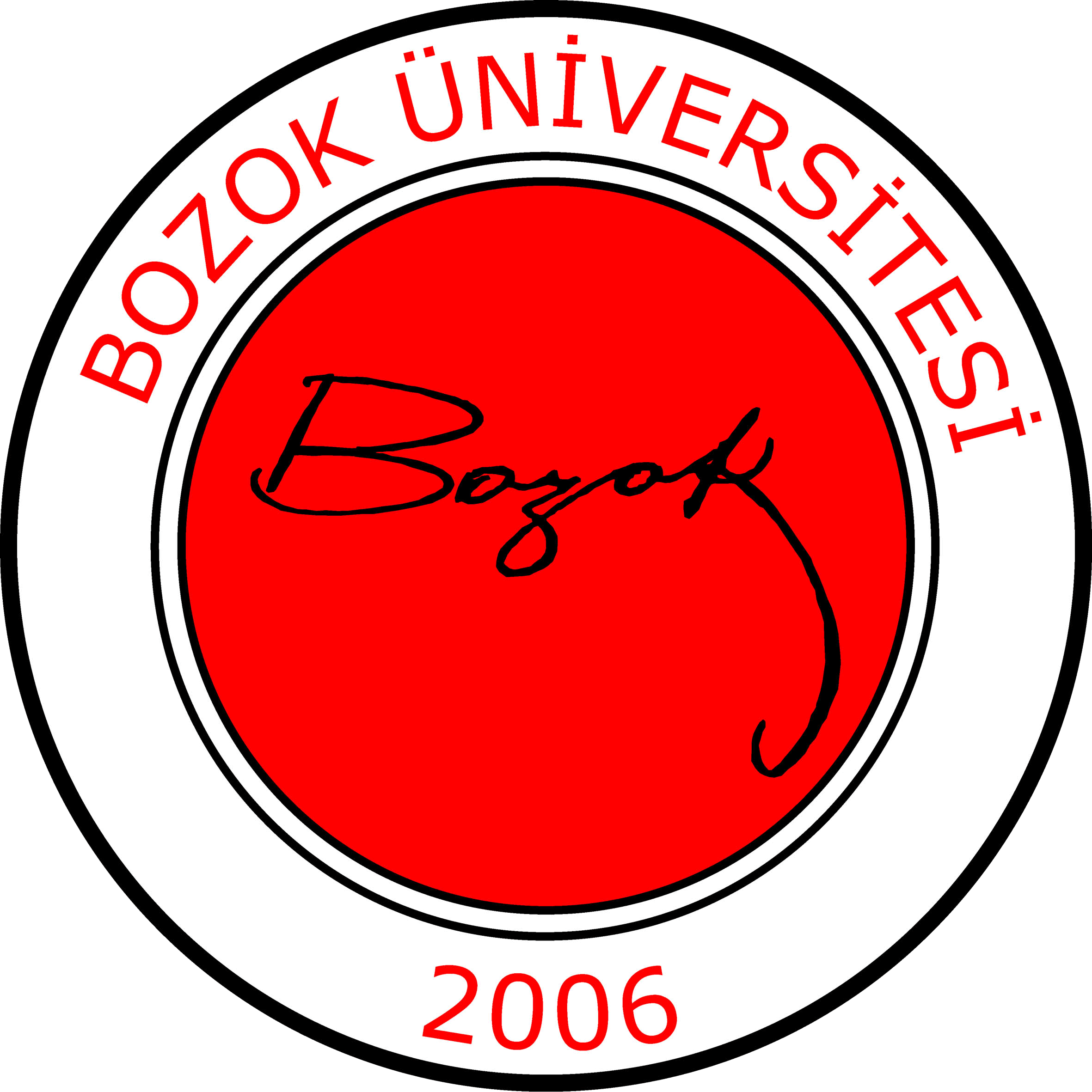
**BOZOK ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**GÖRÜNTÜ İŞLEME İLE MAKİNE ÖĞRENME TEKNİKLERİ KULLANARAK DUYGU DURUM ANALİZİ**

**BİTİRME ÖDEVİ**

**HATİCE NUR TUNCER**

**MUHİTTİN DARI**

**2019-2020 BAHAR DÖNEMİ**

**BOZOK ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**GÖRÜNTÜ İŞLEME İLE MAKİNE ÖĞRENME TEKNİKLERİ KULLANARAK DUYGU DURUM ANALİZİ**

**BİTİRME ÖDEVİ**

**HATİCE NUR TUNCER**

**MUHİTTİN DARI**

**2019-2020 BAHAR DÖNEMİ**

# ÖNSÖZ

Çalışmamız boyunca ilgi ve desteğini esirgemeyen

Sayın Emre ÖLMEZ’e teşekkürlerimizi sunarız.

**Hatice Nur TUNCER**

**Muhittin DARI**

# İÇİNDEKİLER

[ÖNSÖZ I](#_Toc42802100)

[İÇİNDEKİLER II](#_Toc42802101)

[FİGÜR LİSTESİ IV](#_Toc42802102)

[TABLO LİSTESİ V](#_Toc42802103)

[KISALTMA LİSTESİ V](#_Toc42802104)

[ÖZET VI](#_Toc42802105)

[SUMMARY VII](#_Toc42802106)

[GİRİŞ 1](#_Toc42802107)

[1. DİJİTAL GÖRÜNTÜ 1](#_Toc42802108)

[2. MAKİNE ÖĞRENMESİ 1](#_Toc42802109)

[3. ÖĞRENME ALGORİTMALARI 2](#_Toc42802110)

[3.1. Denetimli Öğrenme ( Supervised Learning ) 2](#_Toc42802111)

[3.2. Denetimsiz Öğrenme ( Unsupervised Learning ) 2](#_Toc42802112)

[3.3. Yarı Denetimli Öğrenme ( Semi-Supervised Learning ) 2](#_Toc42802113)

[3.4. Destekli Öğrenme ( Reinforcement Learning ) 3](#_Toc42802114)

[4. YSA BİLEŞENLERİ 3](#_Toc42802115)

[4.1. Nöronlar 4](#_Toc42802116)

[4.2. Bağlantılar Ve Ağırlık Değerleri 4](#_Toc42802117)

[4.3. Yayılım Fonksiyonu 4](#_Toc42802118)

[4.4. Yapay Sinirlerin Organizasyonu 5](#_Toc42802119)

[4.5. Hiper Parametre 5](#_Toc42802120)

[4.6. Öğrenme 5](#_Toc42802121)

[4.7. Öğrenme Oranı 6](#_Toc42802122)

[4.8. Geri Yayılım 6](#_Toc42802123)

[4.9. Perseptron 6](#_Toc42802124)

[4.10. İleri Beslemeli YSA 7](#_Toc42802125)

[4.11. Konvolüsyonel Sinir Ağları 7](#_Toc42802126)

[KULLANILAN ARAÇ VE YÖNTEM 11](#_Toc42802127)

[5. VERİ SETİ 11](#_Toc42802128)

[5.1. Fer2013 Veri Seti 11](#_Toc42802129)

[6. PROGRAMLAMA DİLİ 12](#_Toc42802130)

[6.1. Python 12](#_Toc42802131)

[7. KÜTÜPHANE 12](#_Toc42802132)

[7.1. Tensorflow 12](#_Toc42802133)

[7.2. Keras 12](#_Toc42802134)

[TARTIŞMA VE SONUÇ 13](#_Toc42802135)

[8. OLUŞTURULAN SİNİR AĞI MODELİ VE GELİŞİM SÜRECİ 13](#_Toc42802136)

[8.1. Model A 13](#_Toc42802137)

[8.2. Model B 15](#_Toc42802138)

[8.3. Model C 17](#_Toc42802139)

[9. CANLI VİDEO UYGULAMASI 19](#_Toc42802140)

[9.1. VERİLERİN FER2013 CSV’DEN ALINMASI 19](#_Toc42802141)

[9.2. VERİLERİN EĞİTİM VE TEST OLARAK BÖLÜNMESİ 20](#_Toc42802142)

[9.3. MODELİN EĞİTİLMESİ 21](#_Toc42802143)

[9.4. CANLI VİDEO UYGULAMASININ OLUŞTURULMASI 26](#_Toc42802144)

[ÖZGEÇMİŞ 32](#_Toc42802145)

# FİGÜR LİSTESİ

[Figür 1: Yapay Sinir Ağı Örneği 4](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782210)

[Figür 2: Örnek konvolüsyonel sinir ağı. 8](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782211)

[Figür 3: Birinci modelin yapısının görsel ifadesi. 13](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782212)

[Figür 4: Birinci modelin eğitim ve test sırasında doğruluk ve hata değerleri. 14](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782213)

[Figür 5: İkinci modelin yapısının görsel ifadesi 15](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782214)

[Figür 6: İkinci modelin eğitim ve test sırasında doğruluk ve hata değerleri. 16](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782215)

[Figür 7: Üçüncü modelin yapısının görsel ifadesi 17](file:///C:\Users\darim\Desktop\Python%20Projects\Bitirme\Rapor.docx#_Toc41782216)

[Figür 8:Üçüncü modelin eğitim ve test sırasında doğruluk ve hata değerleri. 18](#_Toc41782217)

# TABLO LİSTESİ

[Tablo 1: fer2013 veri seti etiketleri ve karşılıkları 11](#_Toc41770430)

# KISALTMA LİSTESİ

|  |  |
| --- | --- |
| **KISALTMA** | **AÇILIM** |
| GİB | Grafiksel İşlem Birimi |
| MİB | Merkezi İşlem Birimi |
| KSA | Konvolüsyonel Sinir Ağları |
| DÖ | Derin Öğrenme |
| MÖ | Makine Öğrenmesi |
| YSA | Yapay Sinir Ağları |

# ÖZET

Grafiksel işlem birimlerinin birim zamanda işlem kabiliyetlerinde, internet ortamında bulunan veri miktarında ve internet alt yapısının sahip olduğu veri aktarım hızında son yıllarda yaşanan yüksek artışa bağlı olarak derin öğrenme alanı bir hayli yaygınlaştı. Bu çalışmada, derin öğrenme yöntemlerinden konvolüsyonel sinir ağları kullanılarak duygu analizi gerçekleştirme hedeflenmiştir. Sinir ağına girdi olarak insanların yüz ifadelerini barındıran dijital görüntüler verilmiştir. Sistemi eğitmek için, üç kademede yüz ifadelerini analiz edebilme amacı ile, beş yıl önce Uluslararası Makine Öğrenmesi Konferansı’nda (International Conference on Machine Learning) yayınlanan fer2013 veri seti kullanılmıştır. Bu veri seti toplamda 35.887 gri skala , 48x48 boyutunda, her biri etiketli 7 duyguya tabi yüz görüntülerinden oluşmaktadır. Fer2013, eğitim seti olarak 28.709 ve genel test seti olarak 7178 görüntü içermektedir. Mevcut veri seti ile farklı mimarilerde eğitilen konvolüsyonel sinir ağları arasında farklı değerlendirmeler yapılmıştır. Elde edilen sonuçların başarı oranı göz önünde bulunurularak, en başarılı sonucu alabilmek için konvolüsyonel sinir ağları farklı modeller ile inşa edilmiş, analizlerin sonucu paylaşılmıştır. Son modelin uygulanabilirliği, kullanıcının mevcut duygusunu anında geri bildiren bir canlı video uygulamasında tasvir edilmiştir.

# SUMMARY

Due to the high increase in the processing capacity of GPU’s, the amount of data available on the internet and the speed of data transfer of the internet infrastructure, deep learning has become widespread. In this study, it is aimed to perform emotion analysis using convolutional neural networks which is one of the deep learning methods. Digital images containing the facial expressions of people are given as input to the neural network. To train the system, the fer2013 data set that published at the International Conference on Machine Learning five years ago to analyze facial expressions in three levels was used. This data set consists of a total of 35,887 gray scale, 48x48 size, 7 emotion face images, each labeled. Fer2013 contains 28.709 images as training set and 7178 images as general test set. With the available data set, different evaluations have been made between convolutional neural networks trained in different architectures. Considering the success rate of the obtained results, convolutional neural networks were constructed with different models in order to obtain the most successful results and the results of the analyzes were shared. The applicability of the final model is portrayed in a live video application that can instantaneously return the emotion of the user.

# GİRİŞ

Duygu, çeşitli düşünceler, duygular, davranışsal tepkiler ve bir dereceye kadar hoşnut olma veya hoşnutsuzlukla ilişkili kimyasal değişikliklerin meydana getirdiği sinir sistemi ile ilişkili zihinsel bir durumdur. İnsan duygularının analizi ile, elde edilen duygu durumları ve etkenler eşleştirilerek de insanın memnuniyetine yahut hoşnutsuzluğuna sebebiyet veren etkenleri ayırt edebiliriz. Daha iyi yaşam standartları ve hizmet kalitesi için yapılabilecek ilk iş bu etkenleri tespit edip, çıkarlar doğrultusunda modifiye etmektir.

Her ne kadar bu zamana dek primitif bazı teknikler ile müşteri memnuniyeti sağlama ve benzeri amaçlar için insanların duyguları analiz edilmeye çalışılmış olsa da, elde edilen başarı kullanılan tekniklerin verimli olmaması ve tahminlerin tutarlılığının insafına bırakılmış olması nedeni ile bir hayli düşük seviyede kalmıştır.

Sinirsel aktiviteler sonuçu meydana gelen duyguların analizinde kullanılabilecek en mantıklı araç sinirlerin kendileridir. Bu maksat ile yola çıkılarak, yapay sinir ağlarının duygu analizi işleminde ne kadar başarılı sonuçlar verecek bir araç olduğunu kolaylıkla anlayabiliriz.

İnsan yüzlerinin dijital görüntüleri kullanılarak eğitilecek sistem, mutlu ve üzgün olmak üzere görüntüleri iki ayrı duygudan hangisine sahip olduğunu analiz edecektir.

## 1. DİJİTAL GÖRÜNTÜ

Dijital görüntü piksellerden oluşan, gerçek görüntülerin dijital veriler ile temsil edilmiş halidir. Her bir piksel, tek bir görüntü hücresini ifade eder ve dijital görüntünün en küçük bileşenidir. Görüntülerin dijital ortamda temsil edilmesi için geliştirilen yöntemlerden birisi renkli dijital görüntüler için RGB sistemdir. RGB sisteminde her bir piksel 3 adet sayısal değerden oluşur. Rengin bileşenlerini kırmızı, yeşil ve mavi olarak ayırarak her birisi için 8 bit ayrılmıştır, yani 0 ile 255 arasında değer biçilmektedir.

## 2. MAKİNE ÖĞRENMESİ

Makine öğrenmesi; sistemlerin harici olarak programlanmadan deneyimleri üzerinden otomatik olarak öğrenmesini ve gelişmesini sağlayan, yapay zekanın bir uygulama alanıdır. [1] Makine öğrenmesi, veriye erişip kendini eğitme amacı ile kullanabilen bilgisayar programları geliştirmeye odaklanmıştır.

Öğrenme süreci, verilerdeki anlamlı ifadeleri aramak ve sisteme sağladığımız örneklere dayanarak gelecekte daha iyi kararlar almak için örnekler, doğrudan deneyim veya talimat gibi gözlemler veya verilerle başlar. Birincil amaç, bilgisayarların insan müdahalesi veya yardım olmadan otomatik olarak öğrenmesini sağlamak ve eylemleri buna göre ayarlamaktır.

Makine öğrenmesi büyük miktarda veri analizini mümkün kılar. Kârlı fırsatları veya tehlikeli riskleri belirlemek için genellikle daha hızlı, daha doğru sonuçlar verirken, uygun şekilde eğitmek için ek zaman ve kaynak gerektirebilir. Makine öğrenimini yapay zeka ve bilişsel teknolojilerle birleştirmek, büyük hacimli bilgilerin işlenmesinde daha da etkili olabilir. [2]

## 3. ÖĞRENME ALGORİTMALARI

### 3.1. Denetimli Öğrenme ( Supervised Learning )

Etiketli, durumu önceden bildirilen verilerden öğrendiklerini sonradan verilen verilere uygulayarak çıkarımlarda bulunur. Bilinen bir veri setini (öğrenme seti) analiz ile başlar, öğrenme algoritması bir fonksiyon çıkarır ve bu çıkarım sonraki çıktı değerlerini tahmin için kullanılır. Yeterli bir eğitimin ardından sistem, neredeyse karşılaştığı bütün yeni girdiler için kabul edilebilir hata payları ile kestirimlerde bulunabilir. Öğrenme algoritması, oluşturduğu çıktı değerlerini doğruları ile (etiketler aracılığı ile sisteme belirtilen) kıyaslayıp, daha doğru karar alabilmek amacı ile modeli evirme işlemini gerçekleştirir.

### 3.2. Denetimsiz Öğrenme ( Unsupervised Learning )

Eğitim için kullanılan veri seti etiketlere sahip değil ve/veya sınıflandırılmamış ise meydana gelen makine öğrenmesi tekniğidir. Denetimsiz makine öğrenmesi, etiketsiz veriler üzerinden; veri içerisindeki gizli yapıyı, bağlantıları nasıl ortaya çıkarabileceği konusu üzerine çalışır.

### 3.3. Yarı Denetimli Öğrenme ( Semi-Supervised Learning )

Eğitim için hem etiketli hem de etiketsiz verileri kullandığından, hem denetimsiz hem de denetimli öğrenmenin arasında bir yerlerde yer alan tekniktir. Genel olarak etiketli veriler, etiketsiz verilere oranla çok az sayıdadır. Bu metodu kullanan sistemler (problemin doğasına bağlı olarak), öğrenme başarısında gözde görülür artış sergilemektedir. Genellikle elde edilen etiketli veri, onu eğitmek yahut ondan öğrenmek için işe yarar ve alakalı / uygun kaynaklar gerektirdiğinde, yarı denetimli makine öğrenmesi tercih edilir.

### 3.4. Destekli Öğrenme ( Reinforcement Learning )

Destekli makine öğrenmesi; eylemler gerçekleştirerek çevresi ile etkileşime geçen, hataları ve ödülleri (sistemi eğitmek için kullanılacak değerler) keşfeden bir öğrenme metodudur. Deneme ve hata arama, gecikmeli ödül destekli makine öğrenmesinin en önemli karakteristikleridir. Bu yöntem, makinelerin ve yazılım aracılarının performansını en üst düzeye çıkarmak için belirli bir konudaki ideal davranışı otomatik olarak belirlemelerine olanak tanır. Sistemin hangi eylemin en iyi olduğunu öğrenmesi için basit bir ödül geri bildirimi gereklidir; bu destek sinyali (reinforcement signal) olarak bilinir.

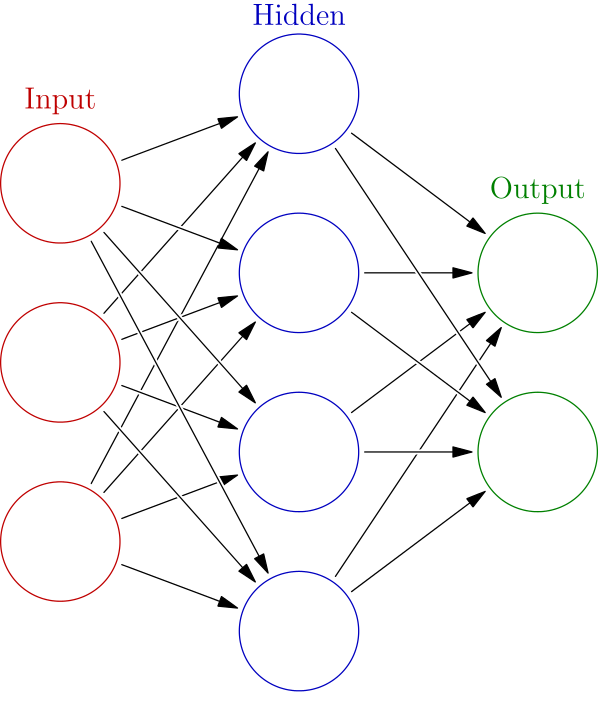
## 4. YSA BİLEŞENLERİ

Yapay sinir ağları (YSA) veya ilişkisel sistemli sistemler, hayvan beyinlerini oluşturan biyolojik sinir ağlarından ilham almış hesaplama sistemleridir. [3] Verilen görevleri yerine getirmek için örnekleri göz önünde bulundurarak öğrenen sistemlerdir. Genelde harhangi bir göreve özel kurallar ile programlanmazlar. Örneğin önceden verilen kedilere ait örnek dijital görüntüleri kullanarak dijital resimlerde kedileri ayırt etme özelliği edinebilirler. Bunu herhangi bir ön bilgi alarak meydana getirmezler. Bunun yerine işlemeleri için verilen görüntüler üzerinden, bu görüntülere ait ortak karakteristikleri üreterek meydana getirirler.

YSA kabaca biyolojik beynin yapısını oluşturmak amacı ile kullanılan, birbirine bağlı yapay nöronlardan oluşur. Her bir bağlantı biyolojik beyindeki sinyallerin iletimi işleminde görev alan sinapsların yapay bir karşılığıdır.

YSA uygulamalarında ‘sinyal’ bir reel sayıdır. Ve her bir nöronun çıktısı, girdilerinin toplamının doğrusal olmayan bir fonksiyona verilip, elde edilen sonuç ile hesaplanır. Nöronlar arası bağlantılara kesit (edge) adı verilir. Nöronlar ve kesitler ‘ağırlık’ adı verilen ve öğrenme süresince ayarlanılabilen bir değere sahiptir. Ağırlık, bağlantıda iletilen sinyalin gücünü artırmak veya azaltmak ile görevlidir. Nöronlar sinyali ancak belirli bir değerin üstünde bir değere sahip olduğunda ileten ‘eşik değeri’ ne sahip olabilir. Tipik olarak nöronlar katmanlar içerisinde toplanır. Farklı katmanlar, girdi değerleri üzerinde farklı dönüşümler uygulayabilir. Sinyaller girdi katmanından çıktı katmanına kadar baştan sona yol alırlar.

YSA’nın temel amacı problemleri insan beyninin çözdüğü yoldan çözmektir. Lakin zamanla ilgi alanı biyolojiden kopup, belirli alandaki görevleri yerine getirmeye doğru yol aldı. YSA'lar, bilgisayarlı görü, konuşma tanıma, makine çevirisi, sosyal ağ filtreleme, oyun tahtası ve video oyunları, tıbbi tanı ve hatta resim çizme gibi geleneksel olarak insanlara rezerve edilmiş faaliyetlerde dahil olmak üzere çeşitli görevlerde kullanılmıştır. [4]



Figür 1: Yapay Sinir Ağı Örneği

### 4.1. Nöronlar

YSA’lar girdi alıp, bir aktivasyon fonksiyonu kullanarak; girdiyi kendi iç durumları (aktivasyon) ve isteğe bağlı bir eşik ile kombin edip, bir çıkış fonksiyonu yardımı ile çıktı üreten yapay nöronların biyolojik kapsamını korudular. İlk katmandaki girdiler dijital görüntü ve dökümanlar gibi harici verilerdir. Nihai çıktılar ise bir görüntüdeki nesneyi tanıma gibi görevleri yerine getirir. Aktivasyon fonksiyonunun önemli özelliği, giriş değerleri değiştikçe yumuşak bir değişim süreci yaratır; yani girdideki ufak değişim çıktıda da ufak değişimlere yol açar. [5]

### 4.2. Bağlantılar Ve Ağırlık Değerleri

Sinir ağı; bir nöronun çıktısının diğer bir nöronun girdisi olabilmesini sağlayan bağlantılara sahiptir. Her bağlantı, kendisinin göreli önemini temsil eden bir ağırlık değerine sahiptir. [6] Bir nöron birden fazla giriş ve çıkış bağlantılarına sahip olabilir. [7]

### 4.3. Yayılım Fonksiyonu

Yayılım fonksiyonu, nöronların girdi değerini belirler. Kendinden önce gelen nöronların sinyallerini ağırlıklarına göre değerlendirip toplamlarını ele alır. [6] Bias (sapma) adı verilen bir değer, ağırlıklı toplamın sonucuna eklenebilir. [8]

f(ç) = (g1\*a1 + g1\*a1 + . . . + gn + an) + S

ç: Çıktı g: Girdi a: Ağırlık Değeri S: Sapma

### 4.4. Yapay Sinirlerin Organizasyonu

Nöronlar, özellikle derin öğrenmede, tipik olarak çoklu katmanlar halinde organize edilirler. Bir katmanın nöronları yalnızca hemen önceki ve hemen sonraki katmanların nöronlarına bağlanır. Harici verileri alan katman giriş katmanıdır. Nihai sonucu üreten katman, çıktı katmanıdır. Aralarında sıfır veya daha fazla gizli katman vardır. Bunlar dışında, katmanlı ve katmansız olan ağlar da kullanılmaktadır. İki katman arasında, çok sayıda bağlantı modelleri mümkündür. Bir katmandaki her nöronun bir sonraki katmandaki her nörona bağlı olduğu ‘tamamen bağlı’ (fully connected) yapıya sahip olabilir. Bir katmandaki bir grup nöronun bir sonraki katmandaki tek bir nörona bağlandığı ve bu katmandaki nöron sayısının azalmasına sebep olan pooling katmanı olabilir.[9] Sadece bu tür bağlantılara sahip nöronlar, yönlendirilmiş bir halkasal olmayan grafik oluşturur ve ‘ileri beslemeli’ ağlar olarak bilinir.[6] Alternatif olarak nöronların, aynı veya önceki katmanlardaki nöronlar arasında bağlantı kurmasına izin veren ağlar tekrarlayan ağlar olarak bilinir.[10]

### 4.5. Hiper Parametre

Hiperparametre, değeri öğrenme sürecine başlamadan önce ayarlanmış bir parametredir. Parametrelerin değerleri öğrenme yoluyla elde edilir. Hiperparametre örnekleri arasında öğrenme oranı, gizli katmanların sayısı ve yığın (batch) büyüklüğü vardır.[11] Bazı hiperparametrelerin değerleri diğer hiperparametrelerin değerlerine bağlı olabilir. Örneğin, bazı katmanların boyutu, genel katman sayısına bağlı olabilir.

### 4.6. Öğrenme

Öğrenme, bir işi daha iyi ele almak için örnek gözlemleri göz önünde bulundurarak ağın uyarlanmasıdır. Öğrenme, sonucun doğruluğunu artırmak için ağın ağırlıklarını (ve isteğe bağlı eşiklerini) ayarlama işlemlerini kapsar. Bu, gözlemlenen hataları en aza indirerek yapılır. Ek gözlemler hata oranını belirgin şekilde azaltamadığı zaman öğrenme işlemi tamamlanmış demektir. Genellikle öğrenmeden sonra bile, hata oranı 0'a ulaşmaz. Öğrendikten sonra, hata oranları çok yüksekse, genellikle ağın olarak yeniden tasarlanması gerekir. Pratik olarak bu, öğrenme sırasında periyodik olarak değerlendirilen bir maliyet fonksiyonu tanımlayarak yapılır. Çıktısı reddetmeye devam ettiği sürece, öğrenme devam eder. Maliyet, genellikle değeri yalnızca yaklaşık olarak tahmin edilebilecek bir istatistik olarak tanımlanır. Çıktılar aslında sayılardan ibarettir, yani hata düşük olduğunda, çıktı (neredeyse bir kedi) ile doğru cevap (kedi) arasındaki fark azdır. Öğrenme, gözlemler arasındaki farkların azaltılmasını amaçlar.[6] Çoğu öğrenme modeli, optimizasyon teorisi ve istatistiksel kestirimin basit bir uygulaması olarak görülebilir.

### 4.7. Öğrenme Oranı

Öğrenme hızı, modelin her gözlemdeki sahip olduğu hataları düzeltmesi amacı ile attığı adımların boyutunu tanımlar. Yüksek bir öğrenme oranı, eğitim süresini kısaltır, ancak modelin düşük bir doğruluğa sahip olmasına sebebiyet verir; düşük bir öğrenme oranı daha uzun sürer, ancak modelin daha yüksek doğruluğa sahip olmasını sağlar. Quickprop gibi optimizasyon yöntemleri öncelikli olarak hatada düşüş yaratma işlemini hızlandırmayı amaçlarken, diğer iyileştirmeler temel olarak güvenilirliği artırmaya çalışır. Alternatif bağlantı ağırlıkları gibi ağ içinde salınım ve kararsızlığa sebebiyet verecek etkenlerin önüne geçmek ve yakınsama oranını arttırmak için, iyileştirmeler duruma göre artan veya azalan uyarlanabilir bir öğrenme oranı kullanır.[12]

### 4.8. Geri Yayılım

Geri yayılma, öğrenme sırasında tespit edilen hatayı düzenlemek ve azaltmak için sinirler arası bağlantıların ağırlıklarını ayarlama yöntemidir. Hata miktarı bağlantılar arasında etkin bir şekilde bölünmüştür. Teknik olarak, geri yayılma, ağırlıklar ile alakalı verilen bir durumla ilişkili maliyet fonksiyonunun eğimini (türevi) hesaplar. Ağırlık güncellemeleri stochastic gradient descent, Extreme Learning Machines, [13] "No-prop" ağlar, [14] geri yayılım olmadan eğitim, [15] "ağırlıksız" ağlar [16][17] ve bağlantısız sinirsel ağlar gibi yöntemlerle yapılabilir.

### 4.9. Perseptron

Makine öğreniminde, perseptron, ikili sınıflandırıcıların denetimli öğrenimi için bir algoritmadır. İkili bir sınıflandırıcı, bir sayılar vektörüyle temsil edilen bir girdinin belirli bir sınıfa ait olup olmadığına karar verebilen bir fonksiyondur.[18] Bir tür doğrusal sınıflandırıcıdır, yani tahminlerini bir dizi ağırlık ile özellik vektörünü birleştiren doğrusal bir tahmin fonksiyonuna dayanarak tahminlerini yapan bir sınıflandırma algoritmasıdır.

### 4.10. İleri Beslemeli YSA

İleriye beslemeli bir yapay sinir ağının düğümleri arasındaki bağlantı bir halkasal döngü oluşturmaz. [6] Bu nedenle, tekrarlayan sinir ağlarından farklıdır.

Feedforward sinir ağı, tasarlanan ilk ve en basit yapay sinir ağı türüdür. [19] Bu ağda, bilgi giriş düğümlerinden, gizli düğümlere (eğer varsa) ve çıkış düğümlerine doğru sadece bir yönde ilerler. Ağ içerisinde döngü veya halkasal yapı yoktur. [6]

#### 4.10.1. Tek katmanlı perseptron

En basit sinir ağı türü, tek bir çıkış düğümlerinin bulunduğu katmandan oluşan tek katmanlı perseptron ağdır; girişler doğrudan bir dizi ağırlık değeri üzerinden çıkışlara beslenir. Ağırlıklar ve girdilerin ürettiği iletilecek sinyal değerlerinin toplamı her düğümde hesaplanır ve değer bir eşiğin üstünde ise (tipik olarak 0) nöron ateşlenir ve aktif olduğunu ifade eden değeri alır (tipik olarak 1); aksi halde devre dışı bırakılmış olduğunu ifade eden değeri alır (tipik olarak -1). Bu tür aktivasyon fonksiyonuna sahip nöronlara aynı zamanda yapay nöronlar veya doğrusal eşik birimleri de denir. Literatürde perseptron terimi genellikle bu birimlerden sadece birinden oluşan ağları ifade eder. 1940'larda Warren McCulloch ve Walter Pitts tarafından da benzer bir nöron tanımlanmıştır.

#### 4.10.2. Çok katmanlı perseptron

Bu ağ sınıfı, genellikle ileri beslemeli bir şekilde birbirine bağlanmış çok sayıda hesaplama biriminden oluşan katmanlardan oluşur. Bir katmandaki her bir nöron, bir sonraki katmanın nöronlarına olan bağlantıları yönlendirmiştir. Birçok uygulamada, bu ağların birimleri aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanır.

### 4.11. Konvolüsyonel Sinir Ağları

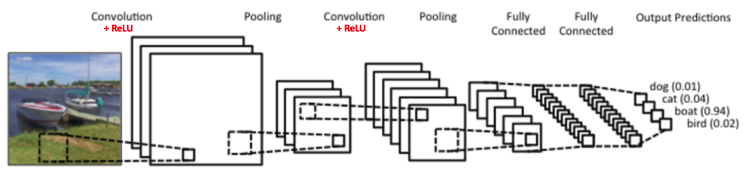
Konvolüsyonel sinir ağları (CNN veya ConvNet), görüntüleri analiz etmek için en sık uygulanan derin sinir ağı sınıfıdır. Ayrıca paylaşımlı-ağırlıklar mimarilerine ve çeviri bağımsızlığı özelliklerine bağlı olarak kaydırma ile değişmez ( zamanla değişmez sistemlerin ayrık karşılığı ) veya uzayda değişmeyen yapay sinir ağları (SIANN) olarak da bilinir. [20] [21] Görüntü ve video tanıma [22], öneri sistemleri, görüntü sınıflandırma, tıbbi görüntü analizi ve doğal dil işleme [23] alanlarında uygulamaları vardır.

#### 4.11.1. LeNet Mimarisi

LeNet, derin öğrenme alanını ilerletmeye yardımcı olan ilk konvolüsyonel sinir ağlarından biriydi. Yann LeCun'un bu öncü çalışması, 1988 yılından bu yana birçok başarılı iterasyonun ardından LeNet5 olarak adlandırıldı [24]. O zamanlar LeNet mimarisi temel olarak posta kodları, rakamlar, vb. karakter tanıma görevleri için kullanıldı.

Aşağıda, LeNet mimarisinin görüntüleri tanımayı nasıl öğrendiğine bir bakış atacağız. Son yıllarda LeNet üzerinde geliştirilen bazı yeni mimariler var ancak LeNet'in ana konseptlerini kullanıyorlar.

Figür 2'deki konvolüsyonel sinir ağının mimarisi, LeNet'in mimarisine benzemektedir ve girdi olarak aldığı görüntüyü; köpek, kedi, tekne veya kuş olmak üzere dört kategoride sınıflandırır. Yukarıdaki şekilde görüldüğü gibi, giriş olarak bir tekne görüntüsü alındığında, ağ dört kategorinin hepsinde tekne için en yüksek olasılığı (0.94) atar. Çıktı katmanındaki tüm olasılıkların toplamı bir olmalıdır (bu yazıda daha sonra açıklanacaktır).



Figür 2: Örnek konvolüsyonel sinir ağı.

Figür 2'de gösterilen konvolüsyonel ağda dört ana işlem vardır:

1. Konvolüsyon
2. Kareselleştirilmiş doğrusal birim uygulama (ReLu)
3. Pooling (birleştirme) ve alt örnekleme
4. Sınıflandırma

Bu operasyonlar, her konvolüsyonel sinir ağının temel yapı taşlarıdır.

# LİTERATÜR TARAMASI

**Nisani (2019).** *“Sentiment analysis using machine learning / Makine öğrenmesini kullanarak duygu analizi”,* Bu araştırmada, duyarlılık polarite analizi çalışılmış ve harici kaynaklardan elde edilen verileri tahmin etmek ve analiz etmek için bir makine öğrenme modeli kullanılmıştır. Naive Bayes yaklaşımı (denetimli makine öğrenme teknikleri) ana sınıflayıcı olarak kullanılır. Bu çalışmadan elde edilen deneysel sonuçlar Naive Bayes yaklaşımının duygu analiz tekniklerinde kullanılan en iyi ve verimli yöntem olduğunu göstermektedir. Verilerin pozitif ve negatif kutuplarının analizi, Naive Bayes sınıflandırıcı, yüksek doğruluk sağlar. Bu yöntem aynı zamanda yüksek kalite üretmek için daha az zaman harcar. [26]

**Demirtaş (2018).** *“Derin öğrenme yöntemleri ile duygu analizi ve aktivasyon fonksiyonlarının karşılaştırılması / Sentiment analysis with deep learning methods and comparison of activation functions”,* Bu çalışmada, sosyal medyada gönderilen mesajların spam mi yoksa ham mı olduğunu anlamak için duygu analizi yapılmaktadır. Duygu analizi bu bağlamda kısa metinleri sınıflandırma problemi olarak da görülebilir. Duygu analizi işlemi için derin öğrenme yöntemi ve klasik sınıflandırma yöntemleri kullanılmış ve sonuçlar karşılaştırılmıştır. Derin öğrenme bağlamında, spam ya da ham mesajların belirlenmesi için kelimeleri vektör uzayında temsil eden kelime gömme (word embedding) ve yinelenen sinir ağları (Recurrent Neural Network/RNN) gibi yöntemler vardır. Çalışmamızda RNN ve kelime gömme yöntemlerini birleştirerek geliştirdiğimiz modelde çeşitli aktivasyon fonksiyonlarının başarımları karşılaştırılmıştır. Klasik sınıflandırma algoritmaları olarak ise SVM (Destek Vektör Makineleri), MNB (Çoklu Nominal Naive Bayes), RF (Rastgele Orman), LR (Lojistik Regresyon), KNN (K Yakın Komşu) kullanılmış ve hem kendi aralarında hem de derin öğrenme yöntemi ile karşılaştırılmıştır. Türkçe ve İngilizce metinlerimizden oluşan veri setlerinde derin öğrenme yöntemi bazı aktivasyon fonksiyonları ile klasik sınıflandırma yöntemlerine göre daha başarılı sonuçlar vermiştir. [27]

**Akışoğlu (2019).** “Predicting economic growth using machine learning techniques and sentiment analysis / Makine öğrenmesi teknikleri ve duygu analizi ile ekonomik büyümenin tahmin edilmesi”, Bu çalışmanın amacı finansal ve ekonomik haberlerden duygu endeksi oluşturmak ve Türkiye'de ekonomik aktiviteyi etkileyen temel ekonomik ve politik olaylarla potansiyel ilişkisini incelemektir. Türkçe'de duygu analizi için etkin bir ekonomik ve finansal sözlük olmadığı için, 2011-2019 dönemi için makine-öğrenme algoritmalarını kullanarak bir duygu endeksi geliştirdik. Bu çalışmada kullanılan veri seti, 2011'den günümüze basında yayınlanan ve dikkatlice belirlenmiş bir kelime grubuna göre seçilen 131.601 haberi içermektedir. Duygu polaritesi belirlenmiş haber veri setini oluşturabilmek amacıyla uzman bir grup sayesinde haberler sınıflandırılmıştır. Duygu endeksinin Türkiye ekonomisi için önemli olayları kapsadığı görülmektedir. Resmi istatistiklerdeki gecikme dikkate alındığında, söz konusu endeks öncü ekonomik gösterge olarak kullanılabilir. Sonraki çalışma konusu olarak, duygu endeksinin ekonomik aktiviteyi açıklayan ekonometrik modellerin açıklayıcı gücünü arttırıp arttırmadığının araştırılması planlanmaktadır. [28]

**Şengür, Abdulkadir. (2016).** *“Derin Aşırı Öğrenme Makinesi ile Yüz Tanıma Deep Extreme Learning Machine for Face Recognition.”* Bu makalede, yerel alıcı alanlar aşırı öğrenme makinesi (YAA-AÖM) ile yüz tanıma gerçekleştirilmiştir. Yüz tanıma, görüntü işleme ve makine öğrenmesi alanlarında ilgi çeken önemli bir uygulamadır. Konu ile alakalı literatürde

birçok metot mevcuttur. Temel bileşen analizi (TBA) tabanlı özyüz ve doğrusal ayrıştırma analizi (DAA) tabanlı Fisheryüz önemli örnekler arasındadır. TBA, yüz görüntülerinin temel bileşenlerini elde edip, adı verilen bu temel bileşenlerin doğrusal bileşimi ile yaklaşık olarak yüz görüntülerini modellemeyi amaçlar. DAA ise aydınlanma ve poz değişiminden daha az etkilenir. Son zamanlarda, büyük veri tabanları ile yüz tanıma, derinyüz kavramını doğurmuştur. Geriye yayılım öğrenme algoritması kullanan Evrişimsel sinir ağlarının (ESA) başarılı yüz tanıma uygulamaları mevcuttur. ESA’nin eğitim sürecinin uzun olması ve eğitim sürecinde yerel bir çözüme takılma ihtimali önemli dezavantajları olarak bilinir. Aşırı öğrenme makinesi (AÖM), geriye yayılım algoritmasının bazı dezavantajlarını gidermek için önerilmiş popüler bir sınıflandırma yöntemidir. AÖM iyi genelleme performansı, aşırı hızlı öğrenme yeteneği ve düşük işlem karmaşıklığı gibi önemli avantajlara sahiptir. Yerel alıcı alanlar (YAA) bilgisinin AÖM içine entegre edildiği YAA-AÖM yapısı, ESA’ya alternatif bir model olarak önerilmiştir. Böylece bu çalışmada, YAA-AÖM ile yüz tanıma gerçekleştirilmiştir. Bu amaçla genişletilmiş Yale B yüz veri seti kullanılmıştır. Gerçekleştirilen deneysel çalışmalar ve karşılaştırmalar YAA-AÖM’nin yüz tanımada başarılı olduğunu göstermiştir. [29]

# KULLANILAN ARAÇ VE YÖNTEM

## 5. VERİ SETİ

### 5.1. Fer2013 Veri Seti

Proje içerisinde veri seti olarak ‘fer2013’ veri seti kullanılmıştır. ‘fer2013’ veri seti üç kademede yüz ifadelerini analiz edebilme amacı ile, beş yıl önce Uluslararası Makine Öğrenmesi Konferansı’nda (International Conference on Machine Learning) yayınlandı.

Açık kaynak veri seti olan fer2013, Pierre-Luc Carrier ve Aaron Courville tarafından yürütülen bir proje için oluşturuldu. Ardından ICML 2013’ten kısa bir süre önce, bir Kaggle (veri seti paylaşım platformu) yarışması için halka açık olarak paylaşıldı.

Bu veri seti toplamda 35.887 gri skala , 48x48 boyutunda, her biri etiketli 7 duyguya tabi yüz görüntülerinden oluşmaktadır.

Yarışma boyunca , eğitim seti olarak 28.709 ve genel test seti olarak 3.589 görüntü paylaşıldı. Geri kalan 3.589 görüntüye sahip özel test seti ise yarışmanın kazananını belirlemek için kullanıldı.

Veri setindeki sayısal duygu etiketlerinin sözel karşılıkları ve her bir duyguya ait görüntü sayısı Tablo 1’de verilmiştir.

|  |  |  |
| --- | --- | --- |
| Etiket | Görüntü Adedi | Duygu |
| 0 | 4593 | Sinirli |
| 1 | 547 | Tiksinmiş |
| 2 | 5121 | Korkmuş |
| 3 | 8989 | Mutlu |
| 4 | 6077 | Üzgün |
| 5 | 4002 | Şaşırmış |
| 6 | 6198 | Doğal |

Tablo 1: fer2013 veri seti etiketleri ve karşılıkları

## 6. PROGRAMLAMA DİLİ

### 6.1. Python

Python yorumlamalı, yüksek seviyeli genel amaçlı bir programlama dilidir. Dizayn edilme felsefesi okunabilirlik üzerine idi. Sahip olduğu dil yapısı ve nesne yönelimli yaklaşımı programcılara temiz ve mantıksal kodlar yazabilmeleri için olanak sağladı. Python dinamik ve otomatik hafıza kalıntısı temizleme sistemine sahip bir dildir. Prosedürel, nesne yönelimli ve fonksiyonel programlama gibi bir çok programlama paradigmasını destekler ve çok geniş bir standart kütüphaneye sahiptir. Yapay zeka alanında da kendini belirgin şekilde ön plana çıkarmış Python; numpy, scipy, tensorflow, matplotlib gibi kütüphaneler ile de desteklenerek matematiksel ve istatistiksel çalışmalarda büyük başarı göstermiştir.

## 7. KÜTÜPHANE

### 7.1. Tensorflow

TensorFlow, belirli işlerde veri akışı ve farklılaştırılabilir programlama fırsatları sunan için ücretsiz ve açık kaynaklı bir yazılım kütüphanesidir. Yapay sinir ağları gibi makine öğrenmesi uygulamalarında kullanılmakta olan TensorFlow, sembolik bir matematik kütüphanesidir. Birden fazla MİB (Merkezi İşlem Birimi) ve GİB (Grafik İşlem Birimi) üzerinde çalışabilmektedir (CUDA ve SYCL eklentileri yardımı ile).

### 7.2. Keras

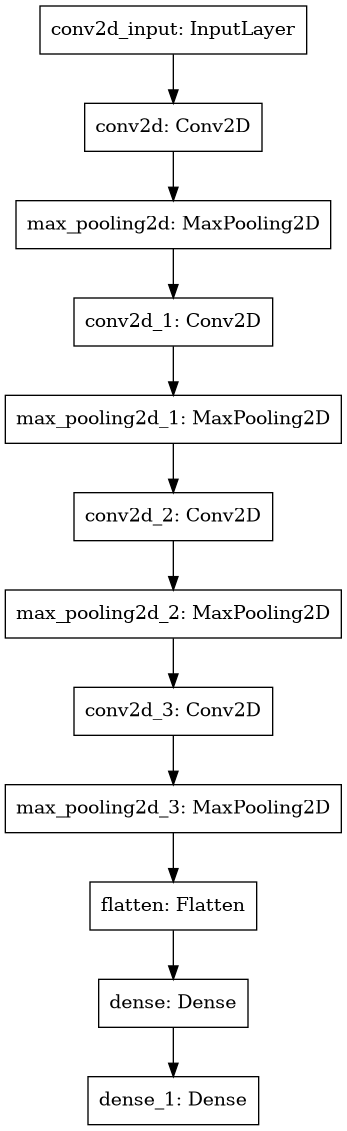
Keras, Python ile yazılmış ve TensorFlow, CNTK veya Theano'nun üzerinde çalışabilen yüksek seviyeli bir sinir ağları uygulama programlama arayüzüdür (API). Hızlıca deneyim edinmeyi amaçlayarak geliştirilmiştir. Kolay ve hızlı prototipleme sağlar (kullanıcı dostu olmak, modülerlik ve genişletilebilirlik aracılığıyla). Konvolüsyonel ağları, yinelemeli ağları ve ikisinin kombinasyonlarını destekler. Hem MİB’nde hem de GİB’nde sorunsuz olarak çalışır.

# TARTIŞMA VE SONUÇ

## 8. OLUŞTURULAN SİNİR AĞI MODELİ VE GELİŞİM SÜRECİ

Hedef, dijital görüntülerin yedi duygu kapsamında incelenmesidir. Bu doğrultuda, ihtiyaç duyulan veriler kullanılan veri setinden ayrıştırılmış ve gerekli modeller oluşturulup test edilmiştir. Kabul edilebilir (%50’nin üzerinde) bir başarı oranı elde edilinceye dek model farklılaştırılmıştır. Bu süreçte oluşturulan modeller ve elde edilen sonuçlar aşağıda verilmiştir.

### 8.1. Model A



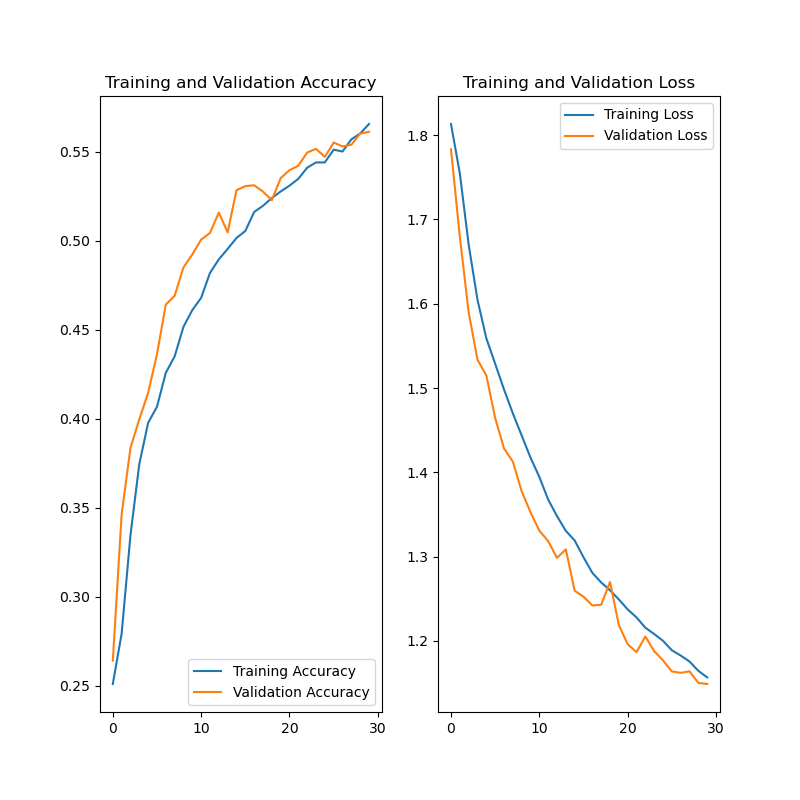
Figür 3: Birinci modelin yapısının görsel ifadesi.

Oluşturulan modelin yapısı giriş katmanından çıkış katmanına aşağıdaki yapıya sahiptir :

1. 48 x 48 x 1 boyutuna sahip iki boyutlu girdi katmanı.
2. 32 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
3. 2 x 2 filtreye sahip pooling katmanı.
4. 64 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
5. 2 x 2 filtreye sahip pooling katmanı.
6. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
7. 2 x 2 filtreye sahip pooling katmanı.
8. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
9. 2 x 2 filtreye sahip pooling katmanı.
10. Düzleştirilmiş katman.
11. 512 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.
12. 7 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.

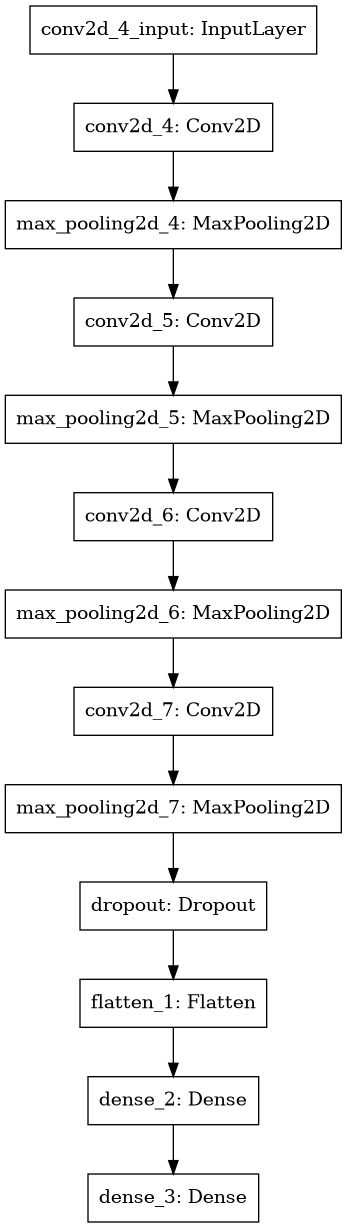
Oluşturulan model ile elde edilen sonuçlar aşağıda verilmiştir :

Elde edilen sonuçlar ile modelin eğitim seti üzerinde ezberleme gerçekleştirdiriği açık olarak görülebilmektedir.



Figür 4: Birinci modelin eğitim ve test sırasında doğruluk ve hata değerleri.

### 8.2. Model B

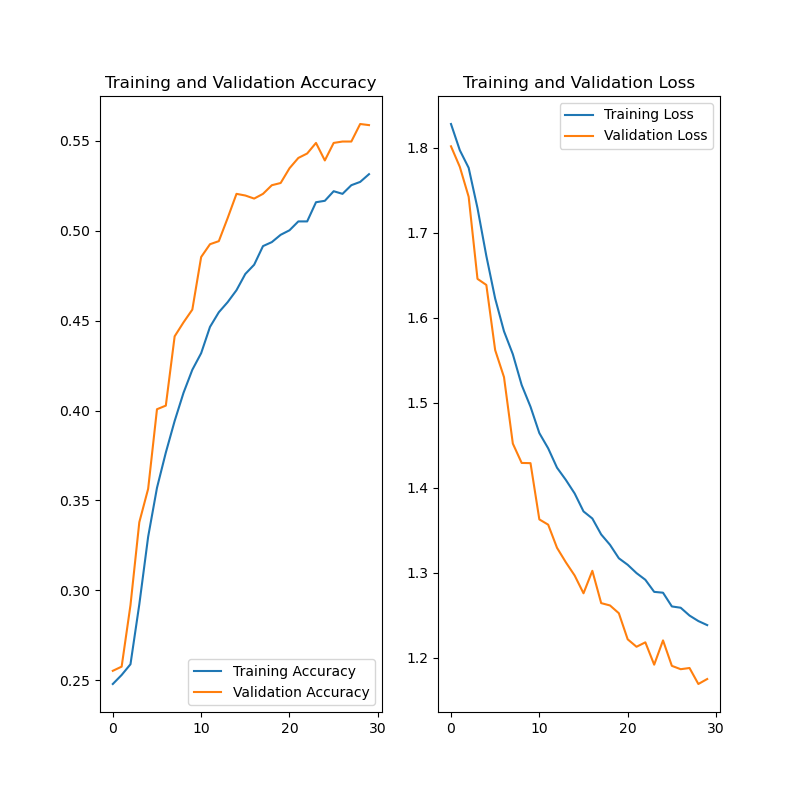


Figür 5: İkinci modelin yapısının görsel ifadesi

Oluşturulan modelin yapısı giriş katmanından çıkış katmanına aşağıdaki yapıya sahiptir :

1. 48 x 48 x 1 boyutuna sahip iki boyutlu girdi katmanı.
2. 32 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
3. 2 x 2 filtreye sahip pooling katmanı.
4. 64 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
5. 2 x 2 filtreye sahip pooling katmanı.
6. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
7. 2 x 2 filtreye sahip pooling katmanı.
8. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
9. 2 x 2 filtreye sahip pooling katmanı.
10. %50 orana sahip dropout katmanı.
11. Düzleştirilmiş katman.
12. 512 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.
13. 7 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.

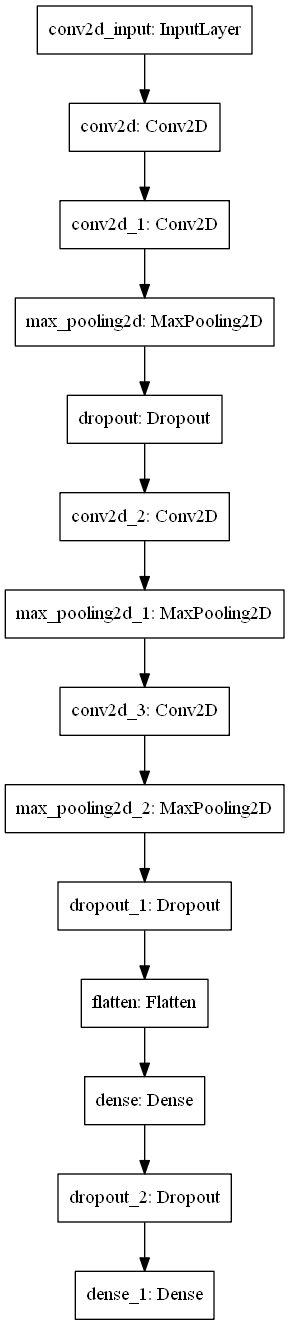
Oluşturulan model ile elde edilen sonuçlar aşağıda verilmiştir :



Figür 6: İkinci modelin eğitim ve test sırasında doğruluk ve hata değerleri.

İlk modelden farklı olarak, 9 ve 10 numaralı katmanlar arasına bir dropout katmanı eklenmiştir. Öğrenme verilerini sisteme besleyen metod parametreleri üzerinde yapılan değişiklikler ile gerçek zamanlı veri artırma (data augmentation) uygulanmış, görüntüler rastgele olarak 40 derece yatay düzlemde çevrilmiş, gerçek görüntülerin 0.2’si oran ile yatay düzlemde rastgele kaydırmalar gerçekleştirilmiş, gerçek görüntülerin 0.2’si oran ile dikey düzlemde rastgele kaydırmalar gerçekleştirilmiş, kaydırma sonucu görüntünün sınırlarının dışına çıkılması ile oluşan boş alanlar ‘en yakın ile doldur’ yaklaşımı ile boş alanlara en yakın piksellerin tekrar edilmesi ile doldurulmuştur.

### 8.3. Model C

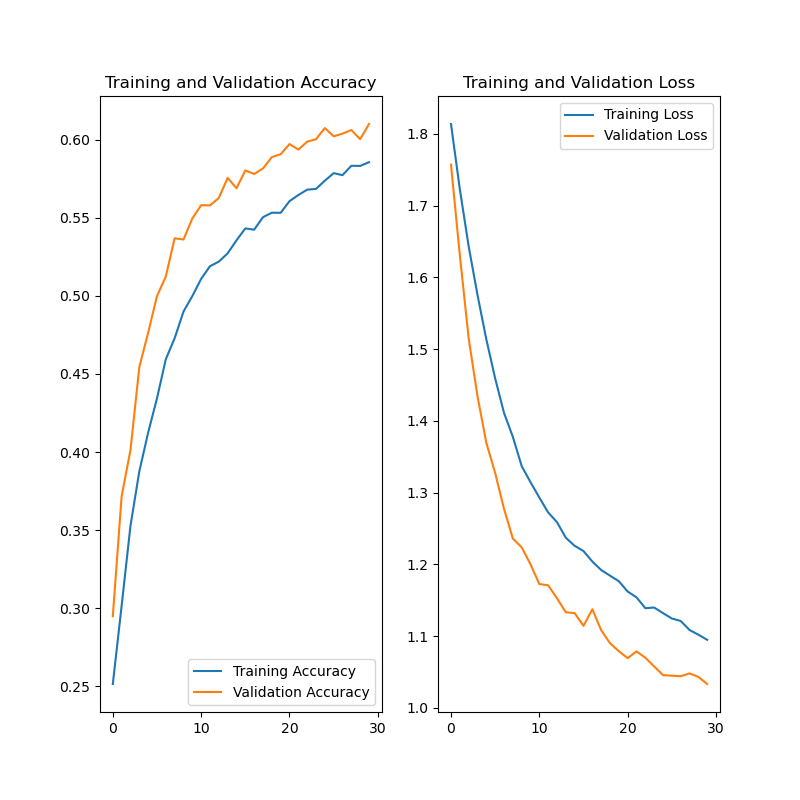


Figür 7: Üçüncü modelin yapısının görsel ifadesi

Oluşturulan modelin yapısı giriş katmanından çıkış katmanına aşağıdaki yapıya sahiptir :

1. 48 x 48 x 1 boyutuna sahip iki boyutlu girdi katmanı.
2. 32 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
3. 64 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
4. 2 x 2 filtreye sahip pooling katmanı.
5. %25 orana sahip dropout katmanı.
6. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
7. 2 x 2 filtreye sahip pooling katmanı.
8. 128 düğümden oluşan, 3x3 filtre kullanan, relu aktivasyon fonksiyonuna sahip 2 boyutlu konvolüsyonel katman.
9. 2 x 2 filtreye sahip pooling katmanı.
10. %25 orana sahip dropout katmanı.
11. Düzleştirilmiş katman.
12. 1024 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.
13. %50 orana sahip dropout katmanı.
14. 7 düğümden oluşan, relu aktivasyon fonksiyonuna sahip tamamen bağlı katman.

Oluşturulan model ile elde edilen sonuçlar aşağıda verilmiştir :



Figür 8:Üçüncü modelin eğitim ve test sırasında doğruluk ve hata değerleri.

Üçüncü modelde ise aynı veri artırım teknikleri kullanılmış, yalnızca model farklılaştırılmıştır. Genel olarak yapılan değişiklik; dropout katmanlarının artırılıp, konvolüsyonel katmanların düğüm sayısının artırılmasından ibarettir. Buna rağmen yaklaşık %10 kadar başarı artışı gözlemlenmektedir.

## 9. CANLI VİDEO UYGULAMASI

Yapılan işlemler sonucunda en başarılı olan ‘Model C‘ modeli oluşturulacak uygulama için kullanılacak model olarak seçilmiştir.

Takip eden adımlarda programın sahip olduğu bölümler aşamalı olarak açıklanmış ve kullanılan kodlar verilmiştir. Her bir adım, kendisinden sonraki adım için gerçekleştirilmesi şart özelliği taşımaktadır.

Uygulamaya “https://github.com/darimuhittin/BozokDuyguAnalizi“ adresi üzerinden erişilebilir.

### 9.1. VERİLERİN FER2013 CSV’DEN ALINMASI

Verilerin fer2013.csv dosyasından alınıp numpy dizileri haline getirilmesi ve dizilerin kayıt işlemi için kullanılan kod aşağıda verilmiştir. İlgili kodlar proje içerisinde ‘HamVeriOlustur.py‘ dosyası içerisinde yer almaktadır. İlgili açıklamalar yorum satırı olarak verilmiştir :

import pandas as pd

import numpy as np

import os

from tensorflow.keras.utils import to\_categorical

# Verilerin kaydedileceği klasör oluştur.

if not os.path.exists('numpyDatas'):

os.makedirs('numpyDatas')

# CSV verisini oku

data = pd.read\_csv('fer2013.csv',',',skiprows=0)

# Verilerin pixel ve etiket bilgilerini ayrıştır

x\_data = data['pixels']

y\_data = data['emotion']

pixels = []

# fer2013 içerisindeki her bir pixel verisi için 48x48x1 boyutunda bir diziye aktar ve bu diziyi pixeller dizisinde biriktir.

for row in x\_data:

arr = row.split(' ')

arr = np.reshape(arr,(48,48,1))#FOR 2D

pixels.append(arr)

# Düzenlenmiş pixel datalarını yeni x\_data olarak güncelle

x\_data = pixels

# Pixeller dizisini 8 bit işaretsiz tamsayı veri tipine sahip numpy dizisine çevir.

x\_data = np.array(x\_data,dtype='uint8')

# Etiket verisini kategorik veriye çevir.

y\_data = to\_categorical(y\_data)

# Etiket adedini yazdır

print("Number of labels : "+str(len(y\_data[0])))

# Diziler numpyDatas içerisine kaydedilir.

np.save('numpyDatas/x\_data', x\_data)

np.save('numpyDatas/y\_data', y\_data)

### 9.2. VERİLERİN EĞİTİM VE TEST OLARAK BÖLÜNMESİ

Verilerin mevcut numpy dizilerinden alınıp, eğitim, test ve doğrulama amaçlı bölünmesi ile ilgi kodlar aşağıda yer alaktadır. Proje içerisinde ‘TrainTestVeriOlustur.py‘ dosyası içerisinde bulunmaktadır. İlgili açıklamalar yorum satırı olarak verilmiştir :

import numpy as np

from sklearn.model\_selection import train\_test\_split

x = np.load('numpyDatas/x\_data.npy')

y = np.load('numpyDatas/y\_data.npy')

# Veri %70 eğitim, %12 test ve %18 doğrulama olarak bölümlenir.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=42)

X\_test, X\_valid, y\_test, y\_valid = train\_test\_split(X\_test, y\_test, test\_size=0.6, random\_state=41)

# Sonradan kullanılmak üzere veriler kaydedilir.

np.save('numpyDatas/x\_train', X\_train)

np.save('numpyDatas/y\_train', y\_train)

np.save('numpyDatas/x\_test', X\_test)

np.save('numpyDatas/y\_test', y\_test)

np.save('numpyDatas/x\_valid', X\_valid)

np.save('numpyDatas/y\_valid', y\_valid)

### 9.3. MODELİN EĞİTİLMESİ

Modelin eğitilmesi, görselleştirilmesi, veri artırım işlemi, ilgili doğruluk ve hata değerlerinin kaydedilmesi işlemleri ile ilgili kodlar aşağıda yer almaktadır. Proje içerisinde ‘ModelEgit.py‘ dosyası içerisinde bulunmaktadır. İlgili açıklamalar yorum satırı olarak verilmiştir :

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.layers import Conv2D, MaxPooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.losses import categorical\_crossentropy

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.utils import plot\_model

import sys

# Eğitim süreci ve model ile ilgili parametreler belirlenir.

num\_features = 32

width, height = 48, 48

num\_labels = 7

batch\_size = 32

epochs = int(sys.argv[1])

learningRate = 0.0001

# Model oluşturulur.

model = Sequential()

model.add(Conv2D(num\_features, (3, 3), activation='relu', input\_shape=(48, 48, 1)))

model.add(Conv2D(num\_features \* 2, (3, 3), activation='relu', input\_shape=(48, 48, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(num\_features\*2\*2, (3, 3), activation='relu', input\_shape=(48, 48, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(num\_features\*2\*2, (3, 3), activation='relu', input\_shape=(48, 48, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(7, activation='softmax'))

# Modelin grafiksel çıktısı alınır

plot\_model(model,'model\_v3.png')

# Model özeti konsol üzerinde görüntülenir

model.summary()

# Hazırlanan numpy verileri alınır.

# Eğitim için (x-y)\_train.npy, test için (x-y)\_valid.npy verileri kullanılacaktır.

# (x-y)\_test.npy verileri iki aşamalı eğitim işlemi için kullanılabilir.

x\_train = np.load('../../numpyDatas/x\_train.npy')

y\_train = np.load('../../numpyDatas/y\_train.npy')

x\_valid = np.load('../../numpyDatas/x\_valid.npy')

y\_valid = np.load('../../numpyDatas/y\_valid.npy')

# Tensorboard log klasörü

log\_dir = "..\\..\\logs\\fit\\v3"

tensorboard\_callback = tf.keras.callbacks.TensorBoard(log\_dir=log\_dir, histogram\_freq=1)

# Eğitim verileri için veri arttırma işlemini sağlayacak keras kütüphanesine ait veri üretici nesnesi oluşturulur.

train\_datagen = ImageDataGenerator(

rescale=1. / 255,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

shear\_range=0.1,

zoom\_range=0.1,

horizontal\_flip=True)

# Test verileri üzerinde oynamaya gerek olmadığından yalnızca uyumluluk için yeniden boyutlandırma işlemi uygulayacak

# veri üretici nesnesi oluşturulur.

test\_datagen = ImageDataGenerator(rescale=1. / 255)

# Veri üretici x\_train verisine adapte edilir.

train\_datagen.fit(x\_train)

# Model için gerekli parametreler oluşturulur.

model.compile(loss=categorical\_crossentropy,

optimizer=Adam(lr=learningRate, decay=1e-6),

metrics=['accuracy'])

# Model eğitilir, eğitim ile ilişkili veri history içerisine kaydedilir.

history = model.fit\_generator(train\_datagen.flow(x\_train, y\_train, batch\_size=batch\_size),

epochs=epochs,

verbose=1,

validation\_data=test\_datagen.flow(x\_valid, y\_valid, batch\_size=batch\_size),

shuffle=True,

callbacks=[tensorboard\_callback])

# Eğitim sürecindeki doğrulama verisi için doğruluk ve hata değerleri alınır.

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(epochs)

# Eğitim süreci doğruluk ve hata değerleri için görsel oluşturulur.

plt.figure(figsize=(8, 8))

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.savefig('./egitimGrafigi\_v3.png')

plt.show()

# Modelin ve ağırlıkların kaydedilmesi.

model.save("model\_v3")

model.save\_weights("model\_weights\_v3.h5")

### 9.4. CANLI VİDEO UYGULAMASININ OLUŞTURULMASI

Kamera üzerinden görüntü alma, yüzün tespit edilip görüntüden ayrılması, ayrılan görüntünün işlenerek model tarafından tahmine tabi tutulması işlemleri ile ilgili kodlar aşağıda yer almaktadır. Proje içerisinde ‘CanliVideoUygulamasi.py‘ dosyası içerisinde bulunmaktadır. İlgili açıklamalar yorum satırı olarak verilmiştir :

import numpy as np

import cv2

import tensorflow as tf

# Kaydedilen model alınır.

model = tf.keras.models.load\_model('model\_v3')

# openCL kullanımı ve gereksiz mesajlar engellenir.

cv2.ocl.setUseOpenCL(False)

# Yüz sınıflandırıcı nesnesi oluşturulur.

faceClassifier = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

# VideoCapture nesnesi oluşturulur. ( 0 varsayılan kamera flag )

videoCapture = cv2.VideoCapture(0)

# Verilen görüntüyü, verilen model ile tahmin işlemine tabi tutacak fonksiyon oluşturulur

def PredictImageWithModel(image, model):

image = np.array(image)

image = np.reshape(image, (48, 48, 1))

image = np.expand\_dims(image, axis=0)

res = model.predict(image)

# print('Tahmin işlemi tamamlandı.')

return res

while True:

# Sıradaki çerçeve okunur.

ret, frame = videoCapture.read()

# Çerçeve alınmış ise

if (ret):

# Yüz tespit et

faces = tuple(faceClassifier.detectMultiScale(frame))

# Yüz bulundu ise

if (len(faces) != 0):

# Her yüz için

for (x, y, w, h) in faces:

# Yüzün bulunduğu kısmı kes.

framePredict = frame[y:y + h, x:x + w]

# 48x48 ve gri skala resim haline getir.

framePredict = cv2.resize(framePredict, (48, 48))

framePredict = cv2.cvtColor(framePredict, cv2.COLOR\_BGR2GRAY)

# Tahmin edilen çerçeveyi görüntüle.

cv2.imshow('Tahmin edilen çerçeve',framePredict)

# Tahmin gerçekleştir.

sonuc = PredictImageWithModel(framePredict, model)

sonucInd = np.argmax(sonuc)

if sonucInd == 0:

sonucString = 'Sinirli'

elif sonucInd == 1:

sonucString = 'Tiksinmis'

elif sonucInd == 2:

sonucString = 'Korkmus'

elif sonucInd == 3:

sonucString = 'Mutlu'

elif sonucInd == 4:

sonucString = 'Uzgun'

elif sonucInd == 5:

sonucString = 'Sasirmis'

elif sonucInd == 6:

sonucString = 'Normal'

# Orijinal frame'de yüz üzerine dörtgen çiz ve gerekli sonuç yazısını ekle.

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.putText(frame, sonucString, (x, y - 5), cv2.FONT\_HERSHEY\_TRIPLEX, 2, (0, 255, 0))

# Her yüz için SON

# Yüz bulundu ise SON

# Sonuç çerçeveyi goster

cv2.imshow('Cam', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

else:

print("Video capture olusturmada hata - (Kamera bulunamamış olabilir.)")

break

# İşlem bittiğinde kaynakları serbest bırak.

videoCapture.release()

cv2.destroyAllWindows()KAYNAKLAR

[1]. Bishop, C. M., Pattern Recognition and Machine Learning,Springer,,2006.

[2]. https://expertsystem.com/machine-learning-definition/.

[3]. Chen, Yung-Yao; Lin, Yu-Hsiu; Kung, Chia-Ching; Chung, Ming-Han; Yen, I.-Hsuan , Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes, Sensors, vol.19, no.9, pp.2047,2019.

[4]. Bethge, Matthias; Ecker, Alexander S.; Gatys, Leon A. , A Neural Algorithm of Artistic Style, , vol., no., pp.,2015.

[5]. , The Machine Learning Dictionary, , vol., no., pp.,2018.

[6]. Zell, Andreas, Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley,Addison-Wesley,,1994.

[7]. Abbod, Maysam F , Application of Artificial Intelligence to the Management of Urological Cancer, The Journal of Urology, vol.178, no.4, pp.1150-1156,2007.

[8]. DAWSONİ, CHRISTIAN W , An artificial neural network approach to rainfall-runoff modelling., Hydrological Sciences Journal., vol.43, no.1, pp.47–66,1998.

[9]: Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber, , 2011

[10]. Miljanovic, Milos , Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction, Indian Journal of Computer and Engineering, vol.3, no.1, pp.,2012.

[11]. Lau, Suki , A Walkthrough of Convolutional Neural Network — Hyperparameter Tuning, Medium, vol., no., pp.,2017.

[12]: Li, Y.; Fu, Y.; Li, H.; Zhang, S. W. , , 2009

[13]. Huang, Guang-Bin; Zhu, Qin-Yu; Siew, Chee-Kheong , Extreme learning machine: theory and applications, Neurocomputing, vol.70, no.1, pp.489–501,2006.

[14]. Widrow, Bernard; et al. , The no-prop algorithm: A new learning algorithm for multilayer neural networks, Neural Networks, vol.37, no., pp.182–188,2013.

[15]. Ollivier, Yann; Charpiat, Guillaume , Training recurrent networks without backtracking, , vol., no., pp.,2015.

[16]. ESANN, 2009.

[17]. Hinton, G. E. , A Practical Guide to Training Restricted Boltzmann Machines Tech. Rep. UTML TR 2010-003. , , vol., no., pp.,2010.

[18]. Freund, Y.; Schapire, R. E. , Large margin classification using the perceptron algorithm, Machine Learning, vol.37, no.3, pp.277–296,1999.

[19]. Schmidhuber, Jürgen , Deep learning in neural networks: An overview, Neural Networks, vol.61, no., pp.85–117,2015.

[20]. Zhang, Wei, Shift-invariant pattern recognition neural network and its optical architecture, Proceedings of Annual Conference of the Japan Society of Applied Physics,1988.

[21]. Zhang, Wei , Parallel distributed processing model with local space-invariant interconnections and its optical architecture, , vol.29, no.32, pp.4790–7,1990.

[22]. van den Oord, Aaron; Dieleman, Sander; Schrauwen, Benjamin , Deep content-based music recommendation, Curran Associates, vol., no., pp.2643–2651,2013.

[23]. Collobert, Ronan; Weston, Jason, , 2008

[24]. https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba.

[25]. https://www.clarifai.com/technology.

[26]. Nisani, Adnan, "Sentiment analysis using machine learning / Makine öğrenmesini kullanarak duygu analizi" , 2019

[27]. Demirtaş, M. Ahmet, "Derin öğrenme yöntemleri ile duygu analizi ve aktivasyon fonksiyonlarının karşılaştırılması / Sentiment analysis with deep learning methods and comparison of activation functions" , 2018

[28]. Akışoğlu, Berkay, "Predicting economic growth using machine learning techniques and sentiment analysis / Makine öğrenmesi teknikleri ve duygu analizi ile ekonomik büyümenin tahmin edilmesi" , 2019

[29]. Şengür, Abdulkadir, "Derin Aşırı Öğrenme Makinesi ile Yüz Tanıma / Deep Extreme Learning Machine for Face Recognition." , 2016

# ÖZGEÇMİŞ

**İsim Soyisim :** Hatice Nur TUNCER

**Doğum Yeri :** Diyarbakır

**Doğum Tarih :** 22.01.1997

**Lise :** İbn-i Sina Anadolu Lisesi Ankara / Sincan

**Lisans :** Bozok Üniversitesi Bilgisayar Mühendisliği N.Ö. 2016 – 2020 ( Devam Ediyor )

**Staj Deneyimi :** ODTÜ Teknokent Bimobil Yazılım

**Etkinlikler :** Bozok Üniversitesi IEEE Yönetim Kurulunda Görev Aldı

**E-Posta :** haticenurtuncer@gmail.com

**İsim Soyisim :** Muhittin DARI

**Doğum Yeri :** Yozgat / Merkez

**Doğum Tarih :** 02.07.1995

**Lise :** Yozgat Lisesi Yozgat / Merkez

**Lisans :** Bozok Üniversitesi Bilgisayar Mühendisliği N.Ö. 2016 – 2020 ( Devam Ediyor )

**E-Posta :** darimuhittin@gmail.com