

**T.C.**

**BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ**

**BUCAK TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ**

**SESİ YAZIYA DÖNÜŞTÜREN ALGORİTMALARIN İNCELENMESİ**

**Ahmet Nusret ASMA**

**Osman Hulusi YILMAZ**

**Ferhat BULGA**

**Denizhan DURSUN**

**BURDUR, 2023**

**T.C.**

**BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ**

**BUCAK TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**SESİ YAZIYA DÖNÜŞTÜREN ALGORİTMALARIN İNCELENMESİ**

**Ahmet Nusret ASMA**

**Osman Hulusi YILMAZ**

**Ferhat BULGA**

**Denizhan DURSUN**

**Danışman: Dr. Ayhan ARISOY**

**BURDUR,2023**

# ÖZET

**Sesi Yazıya Dönüştüren Algoritmaların İncelenmesi**

**Ahmet Nusret ASMA**

**Osman Hulusi YILMAZ**

**Ferhat BULGA**

**Denizhan DURSUN**

**Burdur Mehmet Akif Ersoy Üniversitesi**

**Bucak Teknoloji Fakültesi**

**Yazılım Mühendisliği Bölümü**

**Danışman: Dr. Ayhan ARISOY**

**Ocak 2024**

Kaggle speech recognition dataset kullanılarak, ses tanıma üzerine yapay zeka, derin öğrenme ve doğal dil işleme yöntemlerinin entegrasyonunu detaylı bir şekilde ele alınmıştır. Ses verileri, özellikle LSTM (Uzun Kısa Vadeli Hafıza) algoritmasının LSTM 200 ve LSTM 300 modelleri, Bidirectional algoritması ve GRU (Gated Recurrent Unit) algoritması gibi çeşitli derin öğrenme algoritmaları kullanılarak incelenmiştir. Bu algoritmalar, ses verilerini analiz etmek, öğrenmek ve ses tanıma performansını artırmak amacıyla özel olarak seçilmiştir. Çalışmanın temel hedefi, bu derin öğrenme algoritmalarının ses tanıma başarısını değerlendirmek ve karşılaştırmaktır. Kaggle veri setinden elde edilen sonuçlar, her bir algoritmanın ses tanıma performansına olan etkisini ayrıntılı bir şekilde ortaya koymaktadır. Bu araştırma, ses tanıma teknolojilerindeki gelişmeleri anlamak ve farklı derin öğrenme yöntemlerinin avantajlarını anlamak için önemli bir kaynak sağlamaktadır. Derinlemesine analizler ve karşılaştırmalar, bu algoritmaların ses tanıma uygulamalarındaki etkinliği üzerine önemli bir ışık tutmaktadır.

**Anahtar Kelimeler:** Dil İşleme, Derin Öğrenme, Ses Tanıma

# GİRİŞ

Dünya genelindeki teknolojik gelişmeler, özellikle yapay zeka, derin öğrenme ve ses tanıma konularında önemli ilerlemelere yol açmıştır. Bu ilerlemeler, gelişmekte olan ülkemizde de ses tanıma teknolojilerinin çeşitli alanlarda kullanımını hızlandırmıştır. Ses tanıma, kullanıcı arayüzlerinden sağlık sektörüne, güvenlik sistemlerinden otomotiv endüstrisine kadar birçok alanda çeşitli uygulamalarla karşımıza çıkmaktadır.

Ülkemizde, özellikle sağlık sektöründe, hasta verilerinin kaydedilmesi ve yönetilmesi, ses tanıma teknolojileri ile daha etkili bir hale getirilmektedir. Sağlık profesyonelleri, konuşma tabanlı sistemlerle hastalarla etkileşim kurabilir ve bu sayede hasta bakımı daha kişiselleştirilmiş bir seviyeye taşınabilir. Ayrıca, otomotiv sektöründe de ses tanıma, sürücü asistanları ve araç içi kontrol sistemlerinde kullanılarak sürüş deneyimini geliştirebilir.

Yapay zeka destekli ses tanıma algoritmaları, geniş bir ses verisi yelpazesi üzerinde eğitildiği için doğal dil işleme konusundaki yetenekleri artırarak daha akıcı ve doğal bir etkileşim sunma potansiyeli taşımaktadır. Bu gelişmeler, ülkemizin bu teknolojileri farklı sektörlerde kullanarak daha verimli ve etkili hale getirmesine olanak tanımaktadır. Yapay zeka, derin öğrenme ve ses tanıma kullanılabilecek önemli araçlardır. Bu bağlamda, özellikle Kaggle speech recognition dataset gibi büyük ve çeşitli veri setleri, ses tanıma algoritmalarının eğitilmesi ve optimize edilmesi açısından kritik öneme sahiptir.

Yapay zeka ve derin öğrenme tekniklerinin ses tanıma alanında kullanımı, özellikle LSTM (Uzun Kısa Vadeli Hafıza) algoritması üzerinden geliştirilmiştir. Bu algoritmalar, ses verilerindeki karmaşık desenleri çıkarma yetenekleriyle bilinirler. Ayrıca, LSTM 200 ve LSTM 300 gibi farklı boyutlardaki varyasyonları, ses tanıma performansını artırmak ve modelin yeteneklerini iyileştirmek amacıyla kullanılmaktadır.

Bidirectional algoritması, ses verilerini hem ileri hem de geri yönde işleyerek daha kapsamlı bir anlama kapasitesi sunar. Bu yöntem, sesin zaman içindeki bağlamını daha iyi anlamak ve ses tanıma doğruluğunu artırmak için kullanılır.

GRU (Gated Recurrent Unit) algoritması, bir diğer derin öğrenme yöntemi olarak ses tanıma modellerinde yer alır. GRU, özellikle hafıza hücrelerinin güncellenmesi konusunda gelişmiş bir yapı sunar ve ses verilerini daha etkili bir şekilde işleyebilir.

Bu teknikler, ülkemizdeki ses tanıma uygulamalarının daha doğru, hızlı ve etkili hale gelmesine katkıda bulunarak, yapay zeka destekli ses tanıma alanındaki gelişmeleri hızlandırmaktadır.

# Genel Bilgiler

**2.1**. **Ses Tanıma ve Yapay Zeka Uygulamaları**

Ses tanıma teknolojileri, porfirinlerin moleküler yapısından esinlenerek geliştirilen yapay zeka ve derin öğrenme algoritmalarıyla birleşerek geniş bir uygulama alanına sahiptir. Ses tanıma, ses verilerini işleyerek konuşma tanıma, komutları anlama, doğal dil işleme ve genel olarak kullanıcı arayüzlerini geliştirmek gibi birçok alanda kullanılmaktadır. Kaggle speech recognition dataset gibi büyük veri setleri, ses tanıma algoritmalarının eğitilmesi ve geliştirilmesi için önemli bir kaynak oluşturmaktadır.

**2.2**. **LSTM ve GRU Algoritmaları ile Ses Tanıma**

Uzun Kısa Vadeli Hafıza (LSTM) ve Gated Recurrent Unit (GRU) gibi derin öğrenme algoritmaları, ses tanıma modellerinin geliştirilmesinde önemli bir rol oynamaktadır. LSTM 200 ve LSTM 300 gibi farklı boyutlandırmalar, modelin karmaşıklığını ve ses verilerini daha etkili bir şekilde anlamasını sağlamaktadır. Bidirectional algoritması, ses verilerini hem ileri hem de geri yönde işleyerek daha kapsamlı bir bağlam anlayışı kazandırır.

**2.3**. **Ses Tanıma Verilerinin İşlenmesi ve Analizi**

Ses tanıma uygulamalarında, ses verilerinin işlenmesi ve analizi için çeşitli yöntemler kullanılmaktadır. Tokenizer preprocessing.text ve LabelEncoder preprocessing gibi teknikler, ses verilerini modelleme sürecinde uygun formata getirerek daha etkili bir eğitim sağlar. Librosa, Soundfile, ve Pickle gibi kütüphaneler, ses dosyalarının işlenmesi ve depolanmasında önemli bir rol oynar. Bu süreçler, ses tanıma modellerinin doğruluğunu artırmak ve genel performansını iyileştirmek için önemlidir.

**2.4** **Literatür çalışması**

Bae ve arkadaşları uyarlanabilir MFCC ve derin öğrenmeyi kullanarak gelişmiş bir ses tanıma yöntemi önermişlerdir. Ses tanıma oranının artması için orijinal sesten ses verilerinin düzgün bir şekilde çıkarılması önem taşımaktadır. Ancak gürültüyü kaldırma işlemi sırasında kullanılan algoritmalar ses sinyallerini bozmaktadır. Bae ve arkadaşlarının önerdiği yönetem bu veri kaybını yeni önerilen bir filtre ile azaltmak üzerine kurgulamıştır (Bae vd., 2016)

Ses teli hastalıklarının tespitinde de ses tanıma önemli rol oynamaktadır. Evrişimli sinir ağı modeli kullanarak normal sesi, vokal atrofiyi, tek taraflı ses felcini, organik ses teli lezyonlarını ve adduktor spazmodik disfoniyi ayır etmede Hu ve arkadaşlarının 0.669 doğruluk oranı elde ettikleri görülmüştür (Zhang vd., 2021).

Video konferansların artmasıyla ve yapay zekanın gelişimi ile bu iki sistemin entegresi üzerine çalışan Tanberk ve arkadaşları derin öğrenmeye dayalı transkripsiyon, ve konuşma sentezi yöntemlerine ilişkin en son çalışmaları gözden geçirmişlerdir. Sonrasında konuşmadan metne, metinden konuşmaya ve konuşmadan metinden konuşmaya şeklinde 3 kategoriye ayırarak önceden eğitilmiş iki modelin test senaryolarını analiz etmişlerdir (Tanberk vd., 2021).

Derin öğrenmedeki son gelişmeler, uçtan uca konuşmadan metne çeviri modelinin, konuşma çevirisi alanını yönlendirmek için umut verici bir yaklaşım olduğunu göstermektedir. Bahar ve arkadaşlarının yapmış olduğu çalışmada önceden eğitilmiş modeller kullanılarak bir modelin farklı bileşenlerinin başarılması gibi, ön eğitim değişkenlerini ve bunların Bleu'da %4'e ve Ter‘de %5'e kadar artış sağlayan nihai performans üzerindeki etkilerini de araştırmışlardır (Bahar vd., 2019).

Arzo ve Utku otomatik konuşma tanıma sistemleri için CNN (Convolutional Neural Networks) ve MFCC (Mel-Frequency Cepstral Coefficients) algoritmasını bir araya getirerek bir ses tanıma sistemi geliştirilmiştir. MFCC algoritması, her bir ses sinyalinden benzersiz özellikler çıkararak, bu özellikleri CNN algoritmasına beslemek için kullanılmıştır. Bu sayede, makineler insan sesini daha iyi anlayabilecek ve doğru bir şekilde tanıyabilecektir. Çalışma, Kaggle TensorFlow Konuşma Tanıma Yarışması'ndan elde edilen 64,721 ses dosyasını içeren bir veri seti üzerinde gerçekleştirilmiştir. Bu veri seti, 30 farklı kategoride ses dosyalarını içermekte olup, eğitim, doğrulama ve test setleri olarak ayrılmıştır. Üç farklı model geliştirilmiş ve her bir model, farklı hiperparametre yapılandırmaları ve mimari seçenekleri ile değerlendirilmiştir. Sonuç olarak, en yüksek doğruluk oranı %88.21 olarak belirlenmiştir. Bu çalışma, MFCC'nin özellik çıkarma ve CNN modelinin özellik öğrenme için kullanılmasının, makinelerin insan sesini daha iyi tanımasını sağladığını ve modelin karmaşıklığını azalttığını göstermiştir. (Mahmood, A., & Utku, K. 2021)

# MATERYAL VE YÖNTEM

## Materyal

TensorFlow’un akademik amaçlar ve araştırma için ücretsiz olarak açık kaynak kodlu paylaştığı Speach Recognition isimli veri seti kullanılmıştır.

Veri setinde çeşitli kişilerin okuduğu 30 metnin ses kayıtları ve bunlara karşılık gelen etiketler bulunmaktadır. Kayıtlar .wav formatındadır ve her kayıtta genellikle bir cümle ya da alt paragraf okunmuştur. Süreleri 1-10 saniye arası değişmektedir. Etiketler ise kayıtlardaki söylenen kelimelerin metin halindedir. Farklı konuşmacılara ait binlerce kayıt ve etiket bulunur. Böylece geniş çeşitlilik sağlanmış olur.

## Aletler ve Cihazlar

Çalışmamızda Intel i7 10750H işlemci, GTX 1650ti ekran kartı ve 8GB+8GB (16GB) DDR4 RAM teknolojileri kullanıldı.

## Yöntem

Görünüşe göre çok sayıda ekrana doğru yaklaşıyor olabiliriz. Her gün, yaygın nesnelerin yeni sürümleri, içine yerleştirilmiş wifi ve parlak dokunmatik ekranlarla 'yeniden icat' ediliyor gibi görünmektedir. Ekran bağımlılığımıza umut vadeden bir çözüm ise sesli arayüzlerdir. Ancak, bağımsız üreticiler ve girişimciler için, ücretsiz, açık veri ve kod kullanarak basit bir konuşma algılayıcı oluşturmak zordur. Birçok ses tanıma veri kümesi, üzerlerine bir sinir ağı modeli inşa edilebilmesi için ön işleme gerektirmektedir. Bu konuda yardımcı olmak için TensorFlow, Speech Commands Veri Setlerini 2017 yılında yayınlamıştır. Bu veri seti, binlerce farklı kişi tarafından söylenen 30 kısa kelimenin 65,000 adet bir saniye uzunluğundaki ifadesini içermektedir.

### Kütüphanelerin İçeri Alınması

LSTM (Long Short-Term Memory): RNN (Recurrent Neural Network) mimarisinin bir türüdür. Zaman serileri veya sıralı verilerin işlenmesinde etkilidir. (Yu vd., 2019)

Dense: Tam bağlantılı (fully connected) bir sinir ağı katmanıdır. Tüm giriş birimleri, tüm çıkış birimlerine bağlıdır. (Zhang vd., 2018)

Dropout: Ağdaki aşırı öğrenmeyi azaltmak için kullanılır. Eğitim sırasında rastgele seçilen bir yüzdeyi (dropout oranı) devre dışı bırakır. (Srivastava vd., 2014)

Embedding: Kategorik veriyi temsil etmek için kullanılır. Özellikle metin verilerinde kelime gömme (word embedding) için kullanılır. (Cui vd. 2018)

Masking: Giriş verisinin belirli değerlerini maskeleme amacıyla kullanılır. Modelin belirli giriş değerlerini görmezden gelmesini sağlar. (Samui vd., 2019)

Bidirectional: Çift yönlü RNN (Recurrent Neural Network) katmanıdır. Giriş verisini hem doğru hem de ters yönde işler. (Schuster vd. 1997)

Flatten: Çok boyutlu veriyi düzleştiren bir katmandır. Özellikle evrişimli (convolutional) katmanlardan çıkan veriyi düzleştirmek için kullanılır. (Jin vd. 2014)

SpatialDropout1D, SpatialDropout2D, SpatialDropout3D: Uzaysal dropout katmanlarıdır. Evrişimli sinir ağı modellerinde kullanılır. (Pishahang , M., 2020)

Conv1D, Conv2D: Sırasıyla 1D ve 2D konvolüsyon katmanlarıdır. Evrişimli sinir ağı modellerinde özellik çıkarma için kullanılır. (Prombut vd., 2021)

BatchNormalization: Mini-batch normalizasyonunu gerçekleştiren bir katmandır. Ağdaki eğitimi stabilize etmeye yardımcı olur. (Huang vd., 2018)

MaxPooling1D, MaxPooling2D: Sırasıyla 1D ve 2D maksimum havuzlama katmanlarıdır. Özellik haritalarındaki önemli özellikleri vurgular. (Hou, C.,2020)

Tokenizer: Metni parçalayarak kelime dizinlerine dönüştüren bir sınıftır. Metin verilerini işlemek için kullanılır. (Rust vd., 2020)

Pad\_sequences: Dizi uzunluklarını belirli bir uzunluğa getirmek için kullanılır. Özellikle metin verilerinin işlenmesinde yaygın olarak kullanılır. (Sharma vd., 2020)

LabelEncoder: Kategorik sınıf etiketlerini sayısal değerlere dönüştürmek için kullanılır. (Zhang vd., 2020)

Normalize: Veriyi belirli bir aralığa normalize etmek için kullanılır. (Eck vd., 2009)

Pyplot: Matplotlib kütüphanesinin bir bölümüdür ve grafik çizimi için kullanılır. (Hunt, J. 2023)

EarlyStopping: Eğitim sırasında erken durma işlemi için kullanılır. Belirli bir metriğin gelişimi durduğunda eğitimi sonlandırabilir. (Prechelt, L. 2002)

MinMaxScaler: Veriyi min-max normalizasyonu ile ölçeklemek için kullanılır. (Deepa vd., 2022)

Scipy: WAV dosyalarını okumak ve yazmak için kullanılır. (Bressert, E. 2012)

Numpy: Sayısal hesaplamalar ve çok boyutlu diziler için temel bir kütüphanedir. (Bressert, E. 2012)

Keras: Yapay zeka modelleri oluşturmak ve eğitmek için kullanılan bir yüksek seviyeli bir kütüphanedir. (Chollet, F. 2018)

CSV: CSV dosyaları ile çalışmak için Python'un dahili bir kütüphanesidir. (Thomas vd., 2019)

Os: İşletim sistemi işlemleri için kullanılan bir modüldür. (Gronenschild vd., 2012)

Tqdm: İterasyon sürecini görselleştirmek için kullanılan bir ilerleme çubuğu kütüphanesidir. (da Costa-Luis, C. O. 2019)

Confusion\_matrix, accuracy\_score: Sınıflandırma modelinin performansını değerlendirmek için kullanılan scikit-learn kütüphanesinin işlevleridir. (Testas, A. 2023)

Seaborn: İstatistiksel veri görselleştirmesi için kullanılan bir kütüphanedir. (Bisong vd., 2019)

Matplotlib.pyplot: Matplotlib kütüphanesinin bir bölümüdür ve grafik çizimi için kullanılır. (Yim vd., 2018)

Time: Zamanla ilgili işlemler için Python'un dahili bir kütüphanesidir. (Faouzi vd., 2020)

Tensorflow (tf): Derin öğrenme modelleri oluşturmak ve eğitmek için kullanılan bir açık kaynaklı kütüphanedir. (Developers, T. 2022)

Librosa: Ses işleme için kullanılan bir kütüphanedir. (McFee vd., 2015)

Soundfile: Ses dosyalarını okumak ve yazmak için kullanılan bir kütüphanedir. (Scheibler vd., 2018)

Pickle: Python nesnelerini serileştirmek ve seri durumdan çıkarmak için kullanılan bir kütüphanedir. (Huang vd., 2022)

Kullanılacak olan kütüphanelerin içe aktarılmış durumları Şekil 3.3.1.-’de gösterilmiştir.

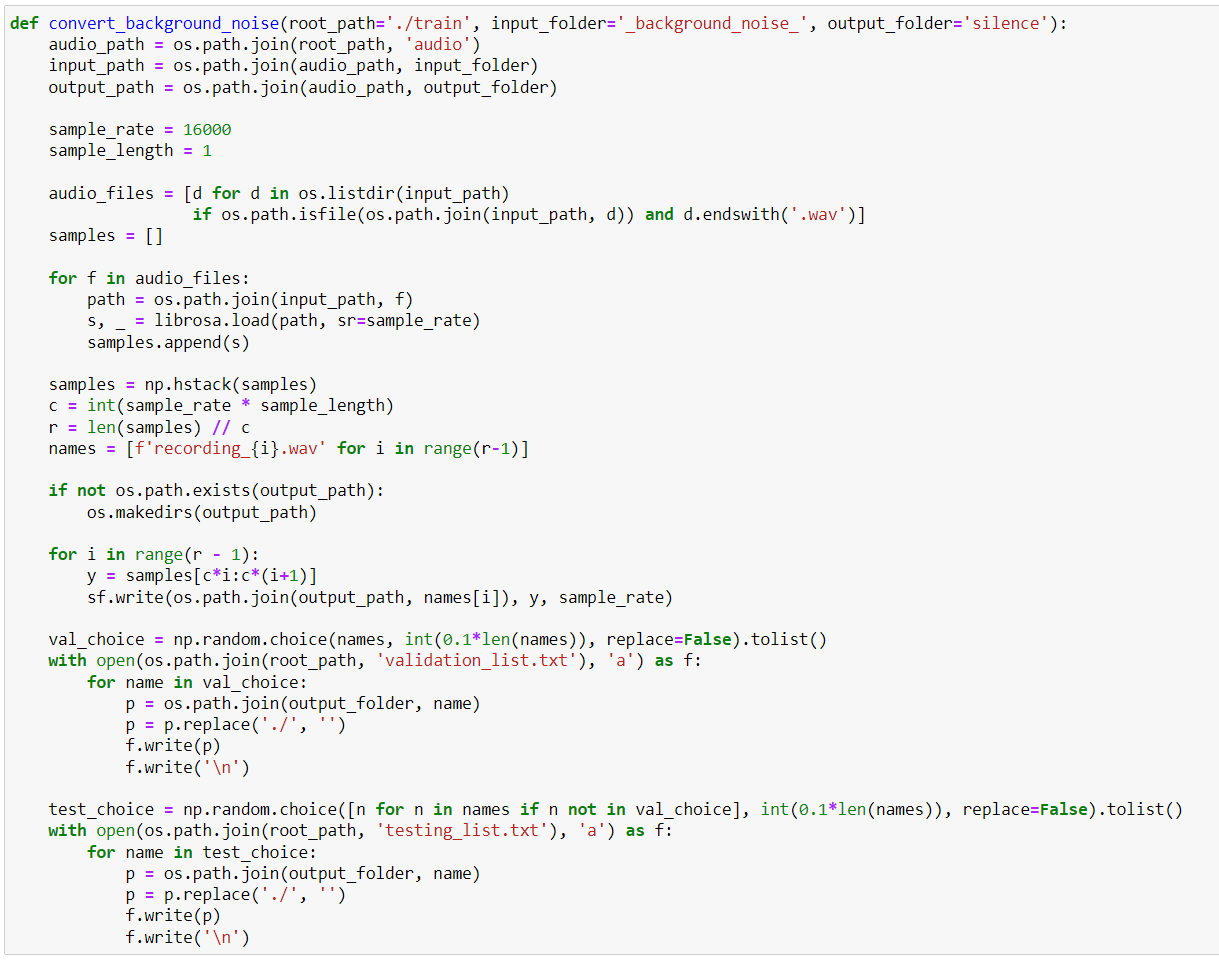
metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil** **3.3.1.** Kullanılacak olan kütüphaneler

### Sessizlik dosyalarının dönüştürülmesi

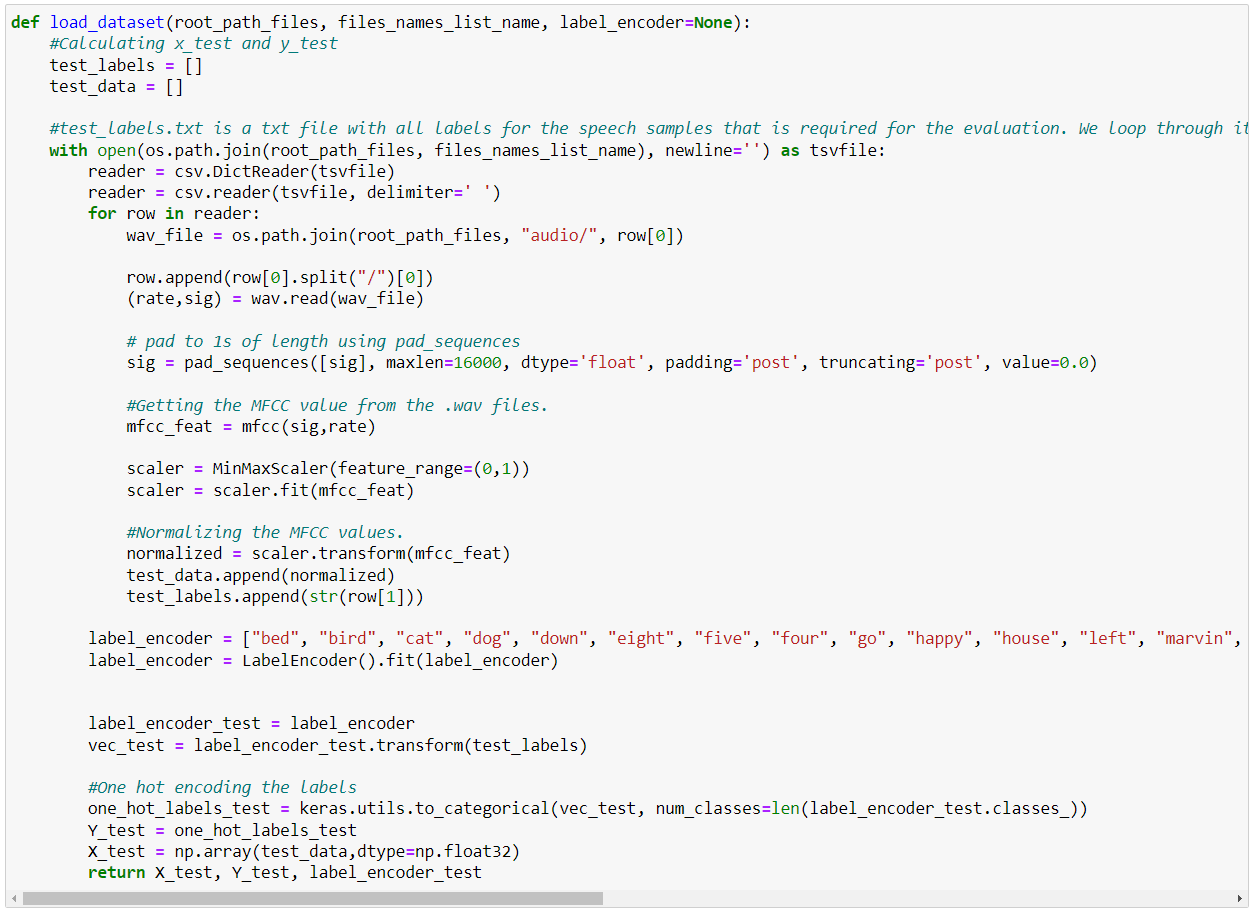
Belirtilen kök dizini altında bulunan " background\_noise " adlı alt dizindeki arka plan gürültüsü içeren ses dosyalarını işleyerek, her biri bir saniye uzunluğunda kesilen ve "silence" adlı bir alt dizine kaydedilen sessizlik dosyalarına dönüştürmektedir. Fonksiyon, ses dosyalarını okuyarak bunları birleştirir, belirli bir uzunluğa bölerek sessizlik dosyalarını oluşturur ve ardından rastgele seçilen bir bölümü doğrulama, geriye kalanı ise test veri seti için kullanılmak üzere ilgili listelere eklemektedir. Bu işlem, özellikle sesli komutları anlamaya yönelik bir modelin eğitimi için arka plan gürültüsünü sessizlikle temsil etmek amacıyla gerçekleştirilmektedir. Sessizlik dosyalarını dönüştüren foksiyon Şekil 3.3.2.’de gösterilmiştir.



**Şekil** **3.3.2.** Sessizlik dosyalarını dönüştüren fonksiyon

**3.3.3 Ses verilerinin işlenerek bir test veri setinin oluşturulması**

Load dataset fonksiyonu, belirtilen kök dizini altındaki ses verilerini işleyerek bir test veri seti oluşturur. Fonksiyon, verilen dosya ismi listesi (files\_names\_list\_name) ve kök dizini (root\_path\_files) üzerinden ses dosyalarını okumaktadır. Her bir ses dosyasının MFCC (Mel-Frequency Cepstral Coefficients) özelliklerini çıkarır ve ardından bu özellikleri normalize etmektedir. Ses dosyalarının etiketleri, her bir dosyanın adından elde edilir ve bu etiketler one-hot encoding ile kodlanmaktadır. Fonksiyon, oluşturulan MFCC özellikleri ve kodlanmış etiketleri içeren X\_test ve Y\_test veri setlerini döndürmektedir. Ayrıca, kullanılan etiketlerin bir kodlayıcısını (label\_encoder\_test) da döndürmektedir. Bu işlev, ses verilerini işleyerek modelin eğitimi veya değerlendirmesi için kullanılabilecek uygun bir test veri seti oluşturmayı amaçlamaktadır. Ses verilerinin işlenerek test veri setini oluşturan load dataset fonksiyonu Şekil 3.3.3.’te gösterilmiştir.



**Şekil 3.3.3.** Load dataset fonksiyonu

**3.3.4 Eğitim veri seti listesinin oluşturulması**

Generate\_train\_txt fonksiyonu, verilen root\_path\_files kök dizini altındaki ses dosyalarını kullanarak bir eğitim veri seti listesi oluşturmaktadır. Fonksiyon, files\_names\_list\_name isimli dosya listesini okuyarak belirli dosyaları atlamaktadır (omit). Daha sonra, kök dizini altındaki "audio" dizinindeki alt dizinlere ve dosyalara bakarak, atlanan dosyalar hariç olmak üzere bir eğitim veri seti listesi oluşturmaktadır. Oluşturulan bu liste, training\_list.txt adlı bir dosyaya yazılmaktadır. Bu dosya, eğitim veri setinde kullanılacak olan ses dosyalarının yollarını içermektedir. Bu işlev, eğitim veri setini oluşturmak ve bu veri setini yönetmek amacıyla kullanılmaktadır. Eğitim veri seti listesinin oluşturan Generate\_train\_txt fonksiyonu Şekil 3.3.4.’ de gösterilmiştir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.4.** Generate\_train\_txt fonksiyonu

**3.3.5 Sesli komutları anlamaya yönelik modelin veri setinin yönetilmesi**

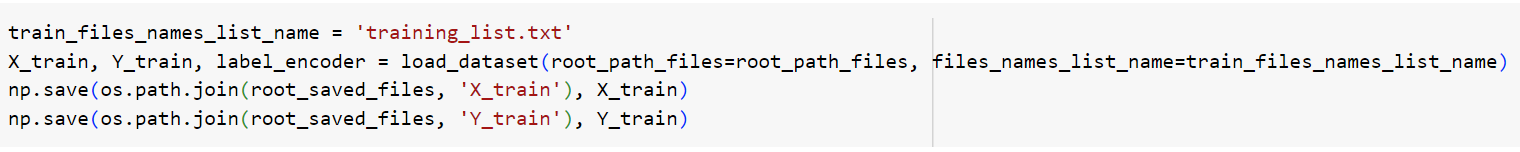
Generate\_train\_txt fonksiyonu, belirtilen root\_path\_files kök dizini altındaki ses dosyalarını kullanarak eğitim veri setini oluşturmaktadır. Fonksiyon, [‘validation\_list.txt’, ‘testing\_list.txt’] adlı iki dosya listesini göz önünde bulundurarak, bu dosyalarda belirtilen ses dosyalarını atlar (omit) ve geriye kalan dosyaları kullanarak bir eğitim veri seti listesi oluşturmaktadır. Bu işlev, özellikle sesli komutları anlamaya yönelik bir modelin eğitimi için kullanılan veri setini yönetmeyi amaçlamaktadır. Oluşturulan eğitim veri seti listesi, training\_list.txt adlı bir dosyaya yazılır ve bu dosya eğitim sürecinde kullanılacak olan ses dosyalarının yollarını içermektedir. Bu sayede, belirtilen dosya listelerine göre eğitim veri seti düzenlenmiş olmaktadır. Generate\_train\_txt fonksiyonu kullanılarak sesli komutları anlamaya yönelik modelin veri setini yönetme amacıyla yazılan kod Şekil 3.3.5.’te gösterilmiştir.



**Şekil 3.3.5.** Generate\_train\_txt fonksiyonu ile veri setini yönetme

**3.3.6 load\_dataset fonksiyonunun kullanılması ve verilerin kaydedilmesi**

Önceki adımlarda oluşturulan eğitim veri setini X\_train ve Y\_train olarak kaydederek disk üzerinde saklamaktadır. load\_dataset fonksiyonu, belirtilen kök dizini (root\_path\_files) altındaki ses dosyalarını kullanarak bir test veri seti oluşturmaktadır. Ardından, np.save fonksiyonlarıyla bu eğitim veri setini Numpy dizileri olarak disk üzerinde kaydetmektedir. X\_train dizisi ses dosyalarının özelliklerini (MFCC değerleri) içerirken, Y\_train dizisi ise bu dosyaların etiketlerini (kategorik olarak kodlanmış) içermektedir. Bu adım, eğitim sürecinde bu veri setini tekrar tekrar yüklemek yerine kaydedilen verileri kullanarak model eğitimini hızlandırmak amacıyla gerçekleştirilir. root\_saved\_files değişkeni, kaydedilen verilerin bulunduğu dizini temsil etmektedir. load\_dataset fonksiyonunun kullanılarak X\_train ve Y\_train verilerinin kaydedilmesi Şekil 3.3.6.’da gösterilmiştir.



**Şekil 3.3.6.** Verilerin kaydedilmesinin gösterilmesi

**3.3.7 Veri setinin Numpy dizileri olarak kaydedilmesi**

Önceki adımlarda oluşturulan doğrulama veri setini disk üzerine Numpy dizileri olarak kaydederek depolamaktadır. Bu adım, modelin eğitim sırasında performansını değerlendirmek amacıyla doğrulama veri setini kaydetmeyi sağlamaktadır. np.save fonksiyonları, ses dosyalarının özelliklerini ve etiketlerini içeren veri setini diske kaydederek, eğitim sürecinde bu veri setini tekrar yüklemek yerine kaydedilen verilere daha hızlı bir erişim sağlamaktadır. root\_saved\_files değişkeni, kaydedilen verilerin bulunduğu dizini temsil etmektedir. Eğitilmiş bir modelin genel performansını değerlendirmek için kullanılacak olan test veri setini (X\_test ve Y\_test) Numpy dizileri olarak disk üzerine kaydetmektedir. load\_dataset fonksiyonu, belirtilen kök dizini (root\_path\_files) altındaki ses dosyalarını kullanarak bir test veri seti oluşturmaktadır. np.save fonksiyonlarıyla bu test veri setini Numpy dizileri olarak kaydederek, modelin performansını daha sonra hızlıca değerlendirmek için bu veri setini disk üzerinde saklamaktadır. root\_saved\_files değişkeni, kaydedilen verilerin bulunduğu dizini temsil etmektedir. Veri setini Numpy dizisi olarak kaydeden np.save fonksiyonunun kullanımı Şekil 3.3.7.’de gösterilmiştir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.7.** Verilerin Numpy dizisi olarak kaydedilmesi

**3.3.8 label\_encoder nesnesini saklama**

Eğitim sırasında kullanılan etiket kodlayıcıyı (label\_encoder) diske encoder adlı bir dosya olarak kaydetmektedir. root\_saved\_files değişkeni, kaydedilen verilerin bulunduğu dizini temsil etmektedir. Bu işlem Şekil 3.3.8’de, eğitimde kullanılan etiket kodlamasını daha sonra kullanmak üzere saklamak amacıyla gerçekleştirilmesi gösterilmiştir.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.8.** label\_encoder nesnesinin saklanması

**3.3.9 Numpy dizisinin yüklenmesi**

Daha önce kaydedilen eğitim veri setini diskten yükleyerek Numpy dizileri olarak elde etmektedir. Bu adım Şekil 3.3.9.’da, eğitilmiş bir modelin kullanılması veya eğitim sürecinin devam ettirilmesi için gerekli olan veri setini elde etmek amacıyla gerçekleştirilmektedir. Daha önce kaydedilen test veri setini (X\_test ve Y\_test) diskten yükleyerek Numpy dizileri olarak elde etmektedir. Bu adım, eğitilmiş bir modelin test edilmesi veya performansının değerlendirilmesi için gerekli olan veri setini hazırlamaktadır. Daha önce kaydedilen doğrulama veri setini (X\_valid ve Y\_valid) diskten yükleyerek Numpy dizileri olarak elde etmektedir. Bu adım, eğitilmiş bir modelin performansını değerlendirmek veya modelin genellemesini kontrol etmek için doğrulama veri setini hazırlamaktadır.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.9.** Verilerin Numpy dizisi olarak yüklenmesi

**3.3.10 Ağırlıkların random atanması**

Numpy ve TensorFlow kütüphanelerindeki rastgele sayı üreteçlerinin başlangıç durumunu, verilen new\_random\_seed değeriyle belirlemektedir. Bu sayede, aynı başlangıç durumuna sahip üreteçler, tekrarlanabilir sonuçlar elde etmek için kullanılabilmektedir. Kullanılan update\_seed fonksiyonu Şekil 3.3.10.’da gösterilmiştir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.10.** update\_seed fonksiyonu

**3.3.11 Eğitim ve görselleştirme**

Verilen bir Keras modelini eğitmek için kullanılmaktadır. Eğitim, doğrulama ve test verileri üzerinde modelin performansını değerlendirmektedir. Aynı zamanda, eğitim sürecini görselleştirir, başarı skorlarını hesaplar ve eğitilen modeli kaydetmektedir. Erken durdurma kullanılarak aşırı uyum önlenmektedir. Modeli eğiten ve eğitilen modelin skorlarını grafik olarak çizdiren fonksiyon Şekil 3.3.11.’de gösterilmiştir.

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.11.** modeli eğiten ve sonuçları grafikleştiren train\_model fonksiyonu

**3.3.12 Ağırlıkların güncellenerek eğitimin tekrarlanması**

Belirli bir modeli farklı başlangıç noktalarıyla (seed'lerle) eğitir ve başarı skorlarını kaydetmektedir. Her bir seed için ayrı bir klasörde saklanan başarı skorları, modelin performansını değerlendirmek ve karşılaştırmak amacıyla kullanılmaktadır. Ağırlıkları güncelleyerek train\_model fonksiyonunu çağırarak farklı ağırlıklarla modeli tekrar eğiten repeat\_train fonksiyonu Şekil 3.3.12.’de gösterilmiştir.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.12.** repeat\_train fonksiyonu

**3.3.13 Modelin oluşturulması**

Bir ses komutlarını tanıma modeli tanımlar ve bu modeli farklı başlangıç noktalarıyla eğiterek performansını değerlendirmektedir. Model oluşturulurken keras kütüphanesinde kullanılan bir modelin başlangıcını temsil eden ve sıralı bir şekilde katmanları sinir ağına eklemek için kullanılan Sequential çerçevesi kullanılmıştır. İlk model, iki katmanlı 200 nörona sahip LSTM'leri içermektedir ve LSTM200 şeklinde adlandırılmıştır. İkinci model iki katmanlı 300 nörona sahip LSTM’leri içermektedir ve LSTM300 şeklinde adlandırılmıştır. Üçüncü model iki katmanlı 200 nörona sahip Bidirectional LSTM’leri içermektedir ve BidirectionalLSTM olarak adlandırılmıştır. Sonuncu model iki katmanlı 200 nörona sahip GRU’ları içermektedir ve GRU olarak adlandırılmıştır. İlk katmanlarda return\_sequences ifadesi true olarak işaretlenmiş ve bu sayede ilk katmanda öğrenilen öz nitelikler ikinci katmana aktarılmıştır. İkinci katmandan sonra aşırı öğrenmeyi engellemek amacıyla Dropout katmanı kullanılmıştır ve değeri 0,2 olarak verilmiştir. Çıkış katmanı olarak 31 düğüme sahip ve aktivasyon fonksiyonu softmax olan bir Dense katmanı tanımlanmıştır. Eğitim süreci, 0, 10, 20, 30, 40 olarak belirlenen başlangıç seed'leri kullanılarak gerçekleştirilmektedir. Modelin öğrenme oranı 0.001, grup büyüklüğü 32, devirleri 25 olarak belirlenmiştir. Elde edilen modeller ve başarı skorları ayrı klasörlerde saklanmaktadır. GRU modelinin oluşturulması ve repeat\_train fonksiyonunda çağırılması Şekil 3.3.13.’de gösterilmektedir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3.3.13.** Modelin oluşturulması ve repeat\_train fonksiyonunun çağırılması

# SONUÇ

Bu çalışma LSTM200, LSTM300, Bidirectional LSTM ve GRU derin öğrenme modellerini inceleyerek Speech Recognation veri seti üzerinde modellerin eğitilmesi ve sonuçlarının kıyaslanmasını amaçlamaktadır.

Eğitilen ilk modelde her biri 200 nörona sahip iki LSTM katmanı kullanılmıştır ve LSTM200 şeklinde isimlendirilmiştir. LSTM200 modelinde eğitim ile beraber yapılan validation sonuçları 0.917 çıkarken test verisindeki sonuçları 0.918 olarak gözlenmiştir. LSTM200 modeline ait test verileri karışıklık matrisi Şekil 4.1.’de görülmektedir.

**Şekil 4.1.** LSTM200 modelinin test verileri karışıklık matrisi

Eğitilen ikinci modelde her biri 300 nörona sahip iki LSTM katmanı kullanılmıştır ve LSTM300 şeklinde isimlendirilmiştir. LSTM300 modelinde eğitim ile beraber yapılan validation sonuçları 0.918 çıkarken test verisindeki sonuçları 0.911 olarak gözlenmiştir. LSTM300 modeline ait test verileri karışıklık matrisi Şekil 4.2.’de görülmektedir. LSTM200 ve LSTM300 modellerinin benzer performansa sahip oldukları söylenebilmektedir. Ancak LSTM200 modeli LSTM300 modeline göre biraz daha iyi genelleme yeteneği göstermektedir.

# Şekil 4.2. LSTM300 modelinin test verileri karışıklık matrisi

Eğitilen üçüncü modelde her biri 200 nörona sahip iki Bidirectional LSTM katmanı kullanılmıştır ve BidirectionalLSTM şeklinde isimlendirilmiştir. BidirectionalLSTM modelinde eğitim ile beraber yapılan validation sonuçları 0.917 çıkarken test verisindeki sonuçları 0.913 olarak gözlenmiştir. BidirectionalLSTM modeline ait test verileri karışıklık matrisi Şekil 4.3.’te görülmektedir. BidirectionalLSTM modelinin test verilerinde biraz daha düşük performansa sahip olduğu gözlenmiştir.

**Şekil 4.3.** BidirectionalLSTM modelinin test verileri karışıklık matrisi

Eğitilen son modelde her biri 200 nörona sahip iki GRU katmanı kullanılmıştır ve GRU şeklinde isimlendirilmiştir. GRU modelinde eğitim ile beraber yapılan validation sonuçları 0.992 çıkarken test verisindeki sonuçları 0.994 olarak gözlenmiştir. GRU modeline ait test verileri karışıklık matrisi Şekil 4.4.’te görülmektedir. GRU modelinin test verilerinde biraz daha düşük performansa sahip olduğu gözlenmiştir.

# Şekil 4.3. GRU modelinin test verileri karışıklık matrisi

Bu çalışma, konuşma tanıma veri seti üzerinde eğitilen dört farklı yenilemeli sinir ağı modelini değerlendirdi. İlk model olan LSTM200, iki katmanlı LSTM ile tasarlanmıştı ve her bir katmanda 200 nöron bulunmaktaydı. Eğitim ve test sonuçları sırasıyla %91.7 ve %91.8 doğruluk oranlarına işaret etti. LSTM200 modeli özellikle “bed”, “five”, “go” kelimelerinin ve gürültünün tespitinde zorlandığı görülmektedir. Ayrıca “tree” ile “three”, “zero” ile “sheila”, “no” ile “go”, “bed” ile “cat” ve “off” ile “up” kelimelerini birbirleriyle karıştırdığı gözlenmektedir. Diğer bir model olan LSTM300, her iki katmanda 300 nöron içeriyordu ve %91.8 doğruluk oranı ile eğitim setinde ve %91.1 doğruluk oranı ile test setinde performans sergiledi. LSTM300 modeli özellikle “bird”, “go”, “up”, “two” kelimelerinin ve gürültünün tespitinde zorlandığı görülmektedir. Ayrıca “off” ile “up”, “no” ile “go”, “no” ile “down” ve “bed” ile “bird” kelimelerini birbirleriyle karıştırdığı gözlenmektedir. Üçüncü model Bidirectional LSTM, iki katmanlı olup her bir katmanda 200 nöron içeriyordu; bu modelin eğitim ve test başarıları sırasıyla %91.7 ve %91.3 olarak belirlendi. Bidirectional LSTM modeli özellikle “eight”, “go”, “three”, “up” kelimelerinin ve gürültünün tespitinde zorlandığı görülmektedir. Ayrıca “down” ile “no”, “tree” ile “three”, “no” ile “go” ve “down” ile “go” kelimelerini birbirleriyle karıştırdığı gözlenmektedir. Son model GRU, iki katmanlı GRU içeriyordu ve her bir katmanda 200 nörona sahipti. Bu model, yüksek başarı oranları ile öne çıktı; eğitim setinde %99.2 ve test setinde %99.4 doğruluk elde etti. BidirectionalLSTM modeli özellikle “eight”, “go”, “three”, “up” kelimelerinin ve gürültünün tespitinde zorlandığı görülmektedir. Şekil 4.5.’te görüldüğü gibi GRU modelinin farklı yerlerdeki hata sayıları da oldukça düşük olarak gözlenmiştir. Sonuçlar, GRU modelinin diğer üç modele göre daha yüksek doğruluk oranlarına sahip olduğunu göstermektedir.

**Şekil 4.4.** GRU modelinin test verileri karışıklık matrisi

# KAYNAKLAR

Bae, H. S., Lee, H. J., & Lee, S. G. (2016, June). Voice recognition based on adaptive MFCC and deep learning. In 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA) (pp. 1542-1546). IEEE.

Bisong, E., & Bisong, E. (2019). Matplotlib and seaborn. Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, 151-165.

Bressert, E. (2012). SciPy and NumPy: an overview for developers.

Chollet, F. (2018). Deep learning mit python und keras: das praxis-handbuch vom entwickler der keras-bibliothek. MITP-Verlags GmbH & Co. KG.

Cui, P., Wang, X., Pei, J., & Zhu, W. (2018). A survey on network embedding. IEEE transactions on knowledge and data engineering, 31(5), 833-852.

da Costa-Luis, C. O. (2019). tqdm: A fast, extensible progress meter for python and cli. Journal of Open Source Software, 4(37), 1277.

Deepa, B., & Ramesh, K. (2022). Epileptic seizure detection using deep learning through min max scaler normalization. Int. J. Health Sci, 6, 10981-10996.

Developers, T. (2022). TensorFlow. Zenodo.

Eck, N. J. V., & Waltman, L. (2009). How to normalize cooccurrence data? An analysis of some well‐known similarity measures. Journal of the American society for information science and technology, 60(8), 1635-1651.

Faouzi, J., & Janati, H. (2020). pyts: A python package for time series classification. The Journal of Machine Learning Research, 21(1), 1720-1725.

Gronenschild, E. H., Habets, P., Jacobs, H. I., Mengelers, R., Rozendaal, N., Van Os, J., & Marcelis, M. (2012). The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements. PloS one, 7(6), e38234.

Huang, L., Yang, D., Lang, B., & Deng, J. (2018). Decorrelated batch normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 791-800).

Huang, N. J., Huang, C. J., & Huang, S. K. (2022, December). Pain Pickle: Bypassing Python Restricted Unpickler for Automatic Exploit Generation. In 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) (pp. 1079-1090). IEEE.

Hunt, J. (2023). Graphing with Matplotlib pyplot. In Advanced Guide to Python 3 Programming (pp. 129-149). Cham: Springer International Publishing.

Jin, J., Dundar, A., & Culurciello, E. (2014). Flattened convolutional neural networks for feedforward acceleration. arXiv preprint arXiv:1412.5474.

Mahmo oud, A., & Utku, K. Ö. S. E. (2021). Speech recognition based on convolutional neural networks and MFCC algorithm. Advances in Artificial Intelligence Research, 1(1), 6-12.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference (Vol. 8, pp. 18-25).

Pishahang, M. H. (2020). Deep Learning HUB: a novel graphical programming platform for big machinery data analytics in reliability engineering & predictive maintenance.

Prechelt, L. (2002). Early stopping-but when?. In Neural Networks: Tricks of the trade (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg.

Prombut, N., Waijanya, S., & Promrit, N. (2021, December). Feature extraction technique based on Conv1D and Conv2D network for Thai speech emotion recognition. In 2021 5th International Conference on Natural Language Processing and Information Retrieval (NLPIR) (pp. 54-60).

Pishahang, M. H. (2020). Deep Learning HUB: a novel graphical programming platform for big machinery data analytics in reliability engineering & predictive maintenance.