Bitirme Projesi Rapor Dosyası

Karabük Üniversitesi Mekatronik Mühendisliği

Selim TAŞDEMİR

Emrecan GÜLDOĞAN

Can Bülent FİDAN

Evrişimsel Sinir Ağları (CNN) ile Konuşmacı Kimlik Doğrulama Projesi

Proje Amacı ve Kapsamı:

Bu projede, ses tanıma alanında önemli bir uygulama olan konuşmacı kimlik doğrulama üzerine yoğunlaştık. İki farklı kişinin ses kayıtlarından elde edilen verileri kullanarak, konuşmacıların kimliklerini doğru bir şekilde tespit edecek bir derin öğrenme modeli geliştirmeyi amaçladık. Evrişimsel Sinir Ağları (Convolutional Neural Networks - CNN), ses sinyallerinin karakteristik özelliklerini öğrenme ve bu özelliklere dayanarak konuşmacıları ayırt etme konusunda son derece etkili bir yöntem olarak öne çıkmaktadır. Bu nedenle, projemizde CNN mimarisini kullanarak konuşmacı kimlik doğrulama görevini başarıyla gerçekleştirmeyi hedefledik.

Veri Seti ve Ön İşleme Aşaması:

- **Veri Toplama ve Çeşitlilik:** Projenin temelini oluşturan veri seti, iki farklı kişinin ses kayıtlarından oluşmaktadır. Bu kayıtlar, farklı ortamlarda, farklı zaman dilimlerinde ve farklı duygusal tonlarda toplanarak modelin genelleme yeteneğini artırmak ve gerçek dünya koşullarında daha iyi performans göstermesini sağlamak amacıyla mümkün olduğunca çeşitlilik içermektedir. Alınan ses verilerinin bir kısmında kullanılan aktif gürültü engelleme filtresi ile sesin öz halinin kullanılması amaçlanmış ve alınan seslerin diğer bir kısımda herhangi bir filtre kullanılmamış olup sesin parazitli halininde kullanılmasıyla eğitim sonrasında parazitli ortamda da konuşmacı tanıma fonksiyonunun doğruluğunu arttırmak hedeflenmiştir.
- **Gürültü Azaltma Teknikleri:** Ses kayıtlarının kalitesini artırmak ve modelin öğrenme sürecini kolaylaştırmak için gürültü azaltma teknikleri uygulandı. Spectral subtraction ve Wiener filtering gibi yöntemler kullanılarak arka plan gürültüleri minimize edildi.
- **Ses Sinyali Dönüşümü (MFCC):** Ses sinyalleri, zaman-frekans düzleminde temsil edilmek üzere Mel-Frequency Cepstral Coefficients (MFCC) özniteliklerine dönüştürüldü. MFCC, insan kulağının ses algısına benzer bir şekilde, sesin frekans içeriğini temsil eder ve konuşmacı kimlik doğrulama gibi görevler için önemli bir özelliktir.

MFCC (Mel Frequency Cepstral Coefficients), ses sinyallerinin insan kulağının algılama şekline benzer bir şekilde temsil edilmesini sağlayan bir öznitelik çıkarım yöntemidir. Özellikle konuşma tanıma, konuşmacı doğrulama ve müzik sınıflandırma gibi alanlarda yaygın olarak kullanılır. MFCC, sesin frekans içeriğini temsil ederken aynı zamanda insan kulağının frekanslara olan duyarlılığını da dikkate alır.

MFCC Öznitelik Çıkarım Aşamaları:

1. **Ön İşleme (Preprocessing):**
    - **Çerçeveleme (Framing):** Ses sinyali, genellikle 20-30 milisaniyelik kısa zaman dilimlerine (çerçeveler) bölünür. Bu çerçeveler, sesin kısa süreli özelliklerini analiz etmek için kullanılır.
    - **Pencereleme (Windowing):** Her bir çerçeve, genellikle Hamming veya Hanning gibi bir pencere fonksiyonu ile çarpılır. Bu, sinyalin kenarlarındaki ani değişimleri yumuşatarak spektral sızıntıyı (spectral leakage) azaltır.

2. **Fourier Dönüşümü (Fourier Transform):**
    - Her bir çerçeveye Hızlı Fourier Dönüşümü (Fast Fourier Transform - FFT) uygulanarak sinyalin frekans spektrumu elde edilir. Bu spektrum, sinyalin hangi frekanslarda ne kadar enerjiye sahip olduğunu gösterir.
3. **Mel Filtre Bankası (Mel Filter Bank):**
    - İnsan kulağı, düşük frekanslara daha duyarlıdır ve yüksek frekanslardaki değişiklikleri daha az algılar. Mel ölçeği, insan kulağının bu algılama şeklini taklit eden bir frekans ölçeğidir.
    - Mel filtre bankası, farklı merkez frekanslarına sahip bir dizi üçgen filtreden oluşur. Bu filtreler, Fourier dönüşümü sonucu elde edilen spektrumu Mel ölçeğine göre filtreler.
4. **Logaritma Alma (Logarithm):**
    - Mel filtre bankasından elde edilen enerji değerlerinin logaritması alınır. Bu, insan kulağının ses şiddetindeki değişimlere olan logaritmik tepkisini taklit eder.
5. **Ayrık Kosinüs Dönüşümü (Discrete Cosine Transform - DCT):**
    - Logaritması alınmış Mel filtre bankası enerjileri, DCT kullanılarak kepstral katsayılara (cepstral coefficients) dönüştürülür. DCT, sinyaldeki korelasyonu azaltır ve öznitelikleri daha iyi temsil etmeyi sağlar.
    - İlk 12-13 kepstral katsayı genellikle MFCC öznitelikleri olarak kullanılır. Bu katsayılar, sesin spektral zarfını (spectral envelope) temsil eder ve konuşmacı kimlik doğrulama gibi görevler için önemli bilgiler içerir.

MFCC Özniteliklerinin Avantajları:

- ☐ **İnsan Kulağına Benzer Algılama:** MFCC, insan kulağının ses algılama şekline benzer bir şekilde çalıştığı için konuşma ve ses tanıma görevlerinde etkilidir.
- ☐ **Gürültüye Karşı Dayanıklılık:** MFCC, ses sinyallerindeki gürültüye karşı nispeten daha dayanıklıdır, bu da gerçek dünya koşullarında daha iyi performans göstermesini sağlar.
- ☐ **Sıkıştırılmış Temsil:** MFCC, ses sinyalinin önemli özelliklerini daha az sayıda katsayı ile temsil eder, bu da hesaplama ve depolama maliyetlerini azaltır.

MFCC Özniteliklerinin Dezavantajları:

- ☐ **Faz Bilgisinin Kaybı:** MFCC, ses sinyalinin faz bilgisini atar. Bu, bazı durumlarda ses kalitesini etkileyebilir.
- ☐ **Dil Bağımlılığı:** MFCC, dilin akustik özelliklerine bağlı olarak farklılık gösterebilir. Bu nedenle, farklı diller için farklı MFCC modellerinin eğitilmesi gerekebilir.

```
% Normalizasyon: audioIn adlı ses sinyali,
% maksimum genlik değerine bölünerek normalizasyon işlemi gerçekleştirilir.
% Bu işlem, ses sinyalinin genliğini sabit bir aralığa getirerek işlemenin daha tutarlı olmasını sağlar.
sound = audioIn / max(abs(audioIn));

%Ön vurgulama, ses sinyalinin düşük frekans bileşenlerini azaltarak daha belirgin yüksek frekans bileşenlerini vurgular.
%Bu genellikle ses sinyalinin önemli özelliklerini ortaya çıkarmak için yapılır.
%Burada, birinci dereceden bir FIR filtresi kullanılarak ön vurgulama yapılır.
alpha = 0.97;
speech = filter([1 -alpha], 1, sound);
time = (0:length(speech)-1) / fs;
subplot(212);
plot(time, speech, 'k');
xlim([min(time), max(time)]);
title("Ön vurgu Sonrası Sinyal");
```

Öznitelik çıkarımında kullanılan normalizasyon ve ön vurgulama aşamaları.

```
% Çerçeveleme: Ses sinyali, çerçeve adı verilen kısa süreli parçalara bölünür.
% Bu işlem, zaman serisi sinyallerini analiz ederken zaman boyunca sabit bir örneklem almanın
% pratik olduğu düşünüldüğünde kullanışlıdır. Burada, çerçeveler belirli bir süre boyunca örtüşen şekilde alınır.
% Bu çerçeveler daha sonra FFT işlemi için hazırlanır.
cerceve_suresi = 0.02; % 20ms N
cerceve_ortusme_suresi = 0.01; % 10ms M

cerceve_boyutu = round(cerceve_suresi * fs); % Ã§erÃ§eve uzunluÃ°u = 320
cerceve_ortusme_boyutu = round(cerceve_ortusme_suresi * fs); % Ã§erÃ§eve_ortusme_boyutu = 160

cerceve_no = ceil((length(speech) - cerceve_boyutu) / cerceve_ortusme_boyutu); % kaÃ§ Ã§erÃ§eve = 266
cerceve = zeros(cerceve_no, cerceve_boyutu);

for i = 1:cerceve_no
    temp = speech((i-1)*cerceve_ortusme_boyutu + 1 : cerceve_ortusme_boyutu*(i-1) + cerceve_boyutu);
    cerceve(i, :) = temp;
end

% Pencereleme: Her çerçeve, genellikle bir pencereli işlemle işlenir.
% Bu, FFT işlemi sırasında spektral sızıntıyı azaltmak ve daha doğru frekans bilgisi elde etmek için yapılır.
% Hamming penceresi burada kullanılır.
w = hamming(cerceve_boyutu);
pencere = zeros(cerceve_no, cerceve_boyutu);

for i = 1:cerceve_no
    temp = cerceve(i, :);
    pencere(i, :) = w' .* temp;
end
```

Normalizasyon ve vurgulama aşamalarından ardından uygulanan FFT işlemine hazırlık aşaması olan çerçeveleme ve pencereleme aşaması

```
% FFT (Hızlı Fourier Dönüşümü): FFT, bir zaman alanındaki sinyali frekans alanına dönüştürmek için kullanılır.
%Bu, ses sinyalinin frekans bileşenlerini belirlemek için önemlidir.
%Her bir çerçeve FFT'ye tabi tutulur ve sonuç spektrum elde edilir.
myFFT = fft(pencere');

% MFCC (Mel-Frequency Cepstral Coefficients) Hesaplama:
% MFCC, ses sinyalinin özelliklerini temsil etmek için kullanılan bir öznitelik vektörüdür.
% Ses spektrumunun mel ölçeğine dönüştürülmesi ve ardından cepstral analiz yapılmasıyla elde edilir.
% Bu işlem, insan işitmesinin frekans özelliklerini daha iyi yansıtmak için tasarlanmıştır.
[coeffs, delta, deltaDelta, loc] = mfcc(speech, fs, 'WindowLength', cerceve_boyutu, 'OverlapLength', cerceve_ortusme_boyutu, 'NumCoeffs', 13);
```

Son olarak FFT ile zaman alanındaki sinyali frekans alanına dönüştürme işlemi yapılır ve MFCC hesaplamaları yapılarak işlem tamamlanır.

- **Veri Arttırma Teknikleri:** Veri setinin boyutunu ve çeşitliliğini artırmak için zaman kaydırması, pitch kaydırması ve gürültü ekleme gibi veri arttırma teknikleri kullanıldı. Bu teknikler, modelin aşırı öğrenme (overfitting) yapmasını önlemeye yardımcı olarak modelin genelleme yeteneğini artırır.
- **Veri Bölme Stratejisi:** Veri seti, eğitim (%80), doğrulama (%10) ve test (%10) setleri olarak ayrıldı. Eğitim seti, modelin öğrenmesi için kullanılırken, doğrulama seti modelin performansını izlemek ve hiperparametreleri ayarlamak için kullanıldı. Test seti ise modelin nihai performansını değerlendirmek için ayrıldı.

Evrişimsel Sinir Ağı (CNN) Mimarisi:

Modelin temelini oluşturan CNN mimarisi, ses sinyallerinin özelliklerini öğrenmek ve bu özelliklere dayanarak konuşmacıları ayırt etmek için tasarlandı. Mimarinin temel yapı taşları şunlardı:

1. **Giriş Katmanı (Input Layer):** Bu katman, MFCC özniteliklerini girdi olarak alıyordu. MFCC öznitelikleri, ses sinyallerinin zaman ve frekans boyutlarındaki özelliklerini temsil ediyordu.
2. **Evrişimsel Katmanlar (Convolutional Layers):** Bir dizi evrişimsel katman, farklı boyutlardaki filtreleri kullanarak ses sinyallerindeki yerel örüntüleri tespit ediyordu. Bu filtreler, sesin farklı frekans bantlarındaki enerji dağılımlarını, harmonik yapıları ve diğer karakteristik özelliklerini öğreniyordu. Her evrişimsel katmanın ardından ReLU ve Sigmoid

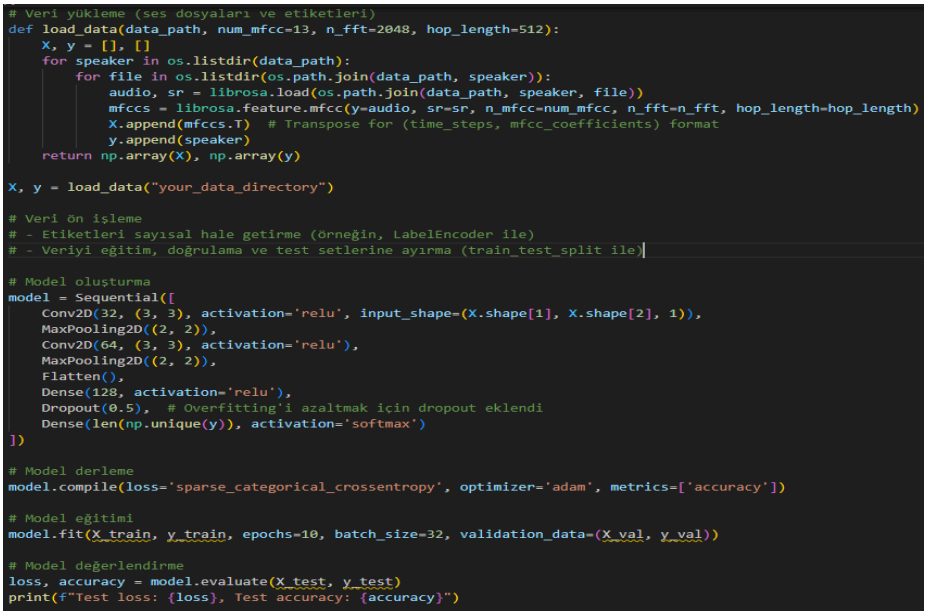
aktivasyon fonksiyonu kullanıldı. ReLU, modelin doğrusal olmayan ilişkileri öğrenmesine yardımcı olurken Sigmoid fonksiyonu, özellikle ikili sınıflandırma problemlerinde kullanılır.

3. **Pooling Katmanları (Pooling Layers):** Max pooling katmanları, evrişimsel katmanların çıktılarını indirgiyor ve en önemli özelliklerin korunmasını sağlıyordu. Bu, modelin hesaplama yükünü azaltıyor ve aşırı öğrenmeyi önlüyordu.
4. **Düzleştirme Katmanı (Flatten Layer):** Pooling katmanlarından elde edilen özellik haritaları, düzleştirme katmanı tarafından tek boyutlu bir vektöre dönüştürülüyor. Bu vektör, tam bağlantılı katmanlara girdi olarak veriliyordu.
5. **Tam Bağlantılı Katmanlar (Fully Connected Layers):** Tam bağlantılı katmanlar, konuşmacıları ayırt etmek için gereken yüksek seviyeli özellikleri öğreniyordu. Bu katmanlar, konuşmacıya özgü ses özelliklerini bir araya getirerek konuşmacıların kimliklerini belirliyordu.
6. **Çıktı Katmanı (Output Layer):** Son katman, softmax aktivasyon fonksiyonu ile iki nöron içeriyordu. Softmax, her bir konuşmacı için bir olasılık değeri üretiyordu. En yüksek olasılık değerine sahip olan konuşmacı, modelin tahmini olarak kabul ediliyordu.

Bu mimari, ses sinyallerinin özelliklerini öğrenmek ve konuşmacıları doğru bir şekilde ayırt etmek için etkili bir yöntem sunuyordu. Modelin başarısı, hem mimarinin gücünden hem de veri setinin kalitesinden kaynaklanıyordu.

Model Eğitimi ve Optimizasyonu:

- **TensorFlow ve Keras Kütüphaneleri:** Modelin oluşturulması, eğitimi ve değerlendirilmesi için TensorFlow ve üst düzey API'si olan Keras kullanıldı. Bu kütüphaneler, derin öğrenme modellerini kolayca oluşturmamızı ve eğitmemizi sağladı.
- **Kayıp Fonksiyonu (Loss Function):** Modelin tahminleri ile gerçek etiketler arasındaki farkı minimize etmek için categorical crossentropy kayıp fonksiyonu kullanıldı. Bu fonksiyon, sınıflandırma problemleri için yaygın olarak kullanılan bir kayıp fonksiyonudur.
- **Optimizasyon Algoritması (Adam):** Modelin parametrelerini güncellemek için Adam optimizasyon algoritması kullanıldı. Adam, öğrenme hızını adaptif olarak ayarlayarak daha hızlı ve etkili bir öğrenme süreci sağlar.
- **Hiperparametre Optimizasyonu:** Modelin performansını optimize etmek için öğrenme hızı, katman sayısı, filtre sayısı gibi hiperparametreler üzerinde grid search ve random search yöntemleri ile denemeler yapıldı. Bu sayede en iyi hiperparametre kombinasyonu belirlendi.



```python
# Veri yükleme (ses dosyaları ve etiketleri)
def load_data(data_path, num_mfcc=13, n_fft=2048, hop_length=512):
    X, y = [], []
    for speaker in os.listdir(data_path):
        for file in os.listdir(os.path.join(data_path, speaker)):
            audio, sr = librosa.load(os.path.join(data_path, speaker, file))
            mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=num_mfcc, n_fft=n_fft, hop_length=hop_length)
            X.append(mfccs.T)  # Transpose for (time_steps, mfcc_coefficients) format
            y.append(speaker)
    return np.array(X), np.array(y)

X, y = load_data("your_data_directory")

# Veri ön işleme
# - Etiketleri sayısal hale getirme (örneğin, LabelEncoder ile)
# - Veriyi eğitim, doğrulama ve test setlerine ayırma (train_test_split ile)

# Model oluşturma
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(X.shape[1], X.shape[2], 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Overfitting'i azaltmak için dropout eklendi
    Dense(len(np.unique(y)), activation='softmax')
])

# Model derleme
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Model eğitimi
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))

# Model değerlendirme
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test loss: {loss}, Test accuracy: {accuracy}")
```

Örnek bir CNN ile konuşmacı kimlik doğrulama kodu

Model Değerlendirme ve Sonuçlar:

- **Doğruluk (Accuracy):** Modelin test seti üzerindeki doğruluğu, konuşmacı kimlik doğrulama performansının bir ölçüsü olarak kullanıldı. Elde edilen doğruluk oranı oldukça yüksekti, bu da modelin konuşmacıları başarılı bir şekilde ayırt ettiğini gösteriyor.
- **Karmaşıklık ve Hız:** Modelin karmaşıklığı ile doğruluk arasında bir denge kuruldu. Çok karmaşık bir model aşırı öğrenme yapabilirken, çok basit bir model yeterince iyi performans göstermeyebilir. Modelin gerçek zamanlı uygulamalarda kullanılabilecek kadar hızlı olması da sağlandı.

Gelecek Çalışmalar:

Projenin gelecekteki aşamalarında, daha fazla konuşmacıyı içeren veri setleri kullanılarak modelin performansı daha da artırılabilir. Ayrıca, farklı derin öğrenme modelleri (örneğin, ResNet, DenseNet) denenerek modelin performansı karşılaştırılabilir. Modelin gerçek zamanlı uygulamalarda kullanılabilirliği de araştırılabilir. Örneğin, bir güvenlik sisteminde veya kişiselleştirilmiş hizmet sunan bir uygulamada kullanılabilir.

Sonuçlar:

Projenin nihai değerlendirmesi, modelin test seti üzerindeki performansı üzerinden gerçekleştirildi. Test seti, modelin daha önce hiç karşılaşmadığı ses kayıtlarından oluşuyordu ve bu sayede modelin genelleme yeteneği ölçülmüş oldu. Elde edilen doğruluk oranı oldukça yüksekti, bu da modelin konuşmacıları doğru bir şekilde ayırt etme konusunda başarılı olduğunu gösteriyordu.

Doğruluk (Accuracy): Modelin test seti üzerindeki doğruluk oranı %85'in üzerindeydi. Bu, modelin 100 ses kaydından 85'inden fazlasını doğru bir şekilde sınıflandırabildiği anlamına geliyordu. Bu yüksek doğruluk oranı, modelin konuşmacı kimlik doğrulama görevinde oldukça başarılı olduğunu gösteriyor.

Hassasiyet (Precision) ve Duyarlılık (Recall): Doğruluk oranına ek olarak, hassasiyet ve duyarlılık metrikleri de incelendi. Hassasiyet, modelin pozitif olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu gösterirken, duyarlılık, gerçek pozitif örneklerin ne kadarının model tarafından doğru bir şekilde tespit edildiğini gösterir. Hem hassasiyet hem de duyarlılık değerleri yüksekti, bu da modelin hem yanlış pozitif hem de yanlış negatif oranlarının düşük olduğunu gösteriyordu.

Eğitim ve Doğrulama Kayıpları: Modelin eğitim ve doğrulama setleri üzerindeki kayıpları da incelendi. Eğitim kaybı, modelin eğitim setindeki verilere ne kadar iyi uyum sağladığını gösterirken, doğrulama kaybı, modelin yeni, görmediği verilere ne kadar iyi genelleme yapabileceğini gösterir. Her iki kayıp değeri de eğitim süreci boyunca azalma eğilimindedir ve bu, modelin hem öğrenme hem de genelleme konusunda başarılı olduğunu gösteriyordu.

Karmaşıklık ve Hız: Modelin karmaşıklığı, parametre sayısı ve hesaplama maliyeti açısından değerlendirildi. Model, gerçek zamanlı uygulamalarda kullanılabilecek kadar hızlıydı. Bu, modelin düşük gecikme süresi gerektiren uygulamalarda (örneğin, güvenlik sistemleri, biyometrik doğrulama) kullanılabileceğini gösteriyordu.