

**SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

**BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI**

**ÖRNEK ÇALIŞMA PLANI**

**3. Rapor**

**Hazırlayanlar:**

**B171210033 - Mertcan Gökmen**

**B181210026 - Gülinsu Özturan**

**B181210068 - Betül Nur Güner**

**İÇERİK**

**Bölüm 1. Projenin Tanımı**

Ses ve komut tanıma sistemleri, günlük hayat içerisinde yer edinmiş ve günlük işleri kolaylaştırmıştır. Bu sistemlerin günümüzdeki en yaygın kullanım alanları; videolarda otomatik altyazı çevirme, akıllı ev asistanları ve sesli yanıt sistemleridir. Bütün bu sistemlerdeki ortak noktalar, ses verilerinin işlenip, anahtar kelimeler dahilinde yönlendirilme yapılmasıdır. Ses tanıma sistemleri kullanılarak reklamların şekillendirilmesi, telefon uygulamalarına hızlı erişim gibi olumlu yönleri olmasına karşın, olumsuz yönleri de bulunmaktadır. Bankaların sesli komutlarla doğrulama yaptığı bu dönemde, kötü niyetli kişilerin bu sistemleri kandırmasıyla güvenlik zaaflarından faydalanılmaktadır.

Bahsedilen bu senaryolarda, İnsan-Makine ilişkisi, Yapay Zeka tabanlı sistemler kullanılarak sağlanmaktadır. Projemizde, genel bir konuşma tanıma modeli oluşturularak, geleneksel ses tanıma işlemlerinden farklı olarak, Derin Öğrenme metotları kullanılarak, sistemin bahsedilen güvenlik açıklıklarının önüne geçmesi amaçlanmaktadır. Bu amaca ulaşmak için, yüksek doğrulukta ve hızlı sonuç üreten bir model üretmek hedeflenmektedir.

**Bölüm 2. Projede Kullanılacak Teknolojiler**

Proje kapsamında, Python programlama dili kullanılacaktır. Python, içerdiği zengin Yapay Zeka kütüphaneleriyle, dünyada en çok kullanılan dillerden birisidir. İçerdiği TensorFlow kütüphanesi yardımıyla Yapay Zeka modeli oluşturulup, eğitim ve test aşamalarından geçecektir. TensorFlow kütüphanesinin bir alternatifi olarak, PyTorch kütüphanesi de bulunmaktadır. PyTorch, alışılmış Python söz dizimine daha yakın olduğu için üzerinde çalışılması daha kolay olmasına rağmen, TensorFlow'un daha fazla kaynağa sahip olmasından ötürü TensorFlow'u tercih edeceğiz.

R programlama dili, Yapay Zeka ve Veri Bilimi alanlarında kullanılan popüler bir alternatiftir. Ancak, ses tanıma sistemleri ve derin öğrenme konularında Python üzerinde daha çok kaynak bulabildiğimiz için R yerine Python programlama dilini tercih edeceğiz.

Ses verilerinin, Derin Öğrenme modeline aktarılmadan önce birtakım işlemlerden geçip hazırlanması gereklidir. Bu işlemler için Librosa kütüphanesi kullanılacaktır. Librosa, müzik ve ses analizi için bir Python paketidir. Müzik bilgi erişim sistemleri oluşturmak için gerekli yapı taşlarını sağlar.

Veri görselleştirme konusunda Python'un en popüler kütüphanelerinden olan Matplotlib kütüphanesinden yararlanılacaktır. Librosa kütüphanesi ile uyumlu çalışarak, ses sinyallerinin görselleştirilmesi sağlanacaktır.

Flask, Web API yazmaya yarayan basit bir Mikro Web Çatısıdır. Uygulamaya çalıştığımız proje için yeterli olmakla birlikte, alternatif olarak Django kullanılabilir. Django, gelişmiş bir Web Geliştirme Çatısı’dır. Flask’tan daha detaylı olmakla birlikte, eğer Flask bizim için yeterli olmaz ise Django’ya geçiş yapacağız.

**Bölüm 3. Yöntemler**

Kullanacağımız modelde, TensorFlow ile Evrişimsel Sinir Ağları (CNN)[1] yöntemi kullanılarak bir sınıflandırma işlemi yapılması hedeflenmektedir. CNN, Yapay Sinir Ağları ve konvolüsyon katmanları içeren bir Derin Öğrenme yöntemidir. Genel olarak resim ve video tanıma, öneri sistemleri, resim sınıflandırma, tıbbi görüntü analizi ve doğal dil işleme alanlarında kullanılır.

Evrişimsel Sinir Ağları içerisindeki Konvolüsyon katmanı, diğer Derin Öğrenme yöntemlerinden ayırt edici bir özelliktir. İçerdiği filtreler yardımıyla verilerin ayırt edici özelliklerini çıkarır ve Pooling işlemi ile, veri boyutunu düşürerek işlem yükünü azaltır. Konvolüsyon katmanlarının sayısı arttıkça, daha belirli örüntüler tanınır, ancak işlem yükü de aynı oranda artar. Bundan dolayı, gerekli test ve başarı ölçümleri yapılarak, en uygun modelin oluşturulması amaçlanır.

Flatten katmanı matris formundaki veriyi düzleştirmek için kullanılır. Bu katmanın görevi son ve en önemli katmana girecek verileri hazırlamaktır. Bu sinir ağındaki veriler Konvolüsyon ve Pooling katmanlarından gelen matrislerin tek boyutlu diziye çevrilmiş halidir.

Pooling katmanının görevi, gösterimin kayma boyutunu ve ağ içindeki parametreleri ve hesaplama sayısını azaltmak içindir. Bu sayede ağdaki uyumsuzluk kontrol edilir. Pooling işleminin en popüleri bizim de kullandığımız max pooling’dir. Average pooling ve L2-norm pooling algoritmaları da vardır ancak biz denemelerimiz sonucu en yüksek verimi max pooling yönteminde aldık. Eğer VAEs veya GANs gibi daha üretken modeller kullansaydık pooling katmanını kullanmamıza gerek kalmayabilirdi ancak proje amacımız üretkenlikle alakalı olmadığı için pooling katmanını kullandık.

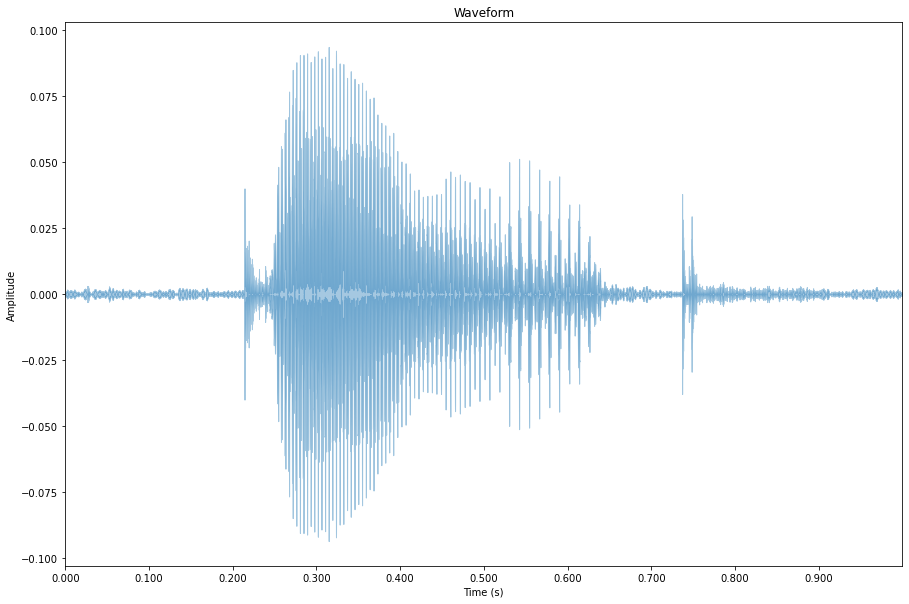
Evrişimsel Sinir Ağıyla ses verisi işlenmesi için, ses verisi belirli bir formatta olmalıdır. Bu formata gelebilmesi için, sese ait öznitelik çıkarımı yapılır. Bunun için Mel Frekans Ölçeği kullanılır. Mel Frekans Ölçeği insan kulağının ses frekanslarındaki değişimi algılayışını göstermektedir.[2] Evrişimsel Sinir Ağları'na ise, ses sinyalinin kısa zamanlı güç spektrumunun Mel ölçeği üzerindeki ifadesi olan MFCC (Mel-Frequency Cepstral Coefficients) formatında verilir. Librosa kütüphanesi, alınan ses verisini bu formata dönüştürür.

Test aşamasına geçildiğinde, bir Web API üzerinden denemeler yapılacaktır. Bunun için Python’un Django ve Flask kütüphanelerinden yararlanılacaktır.

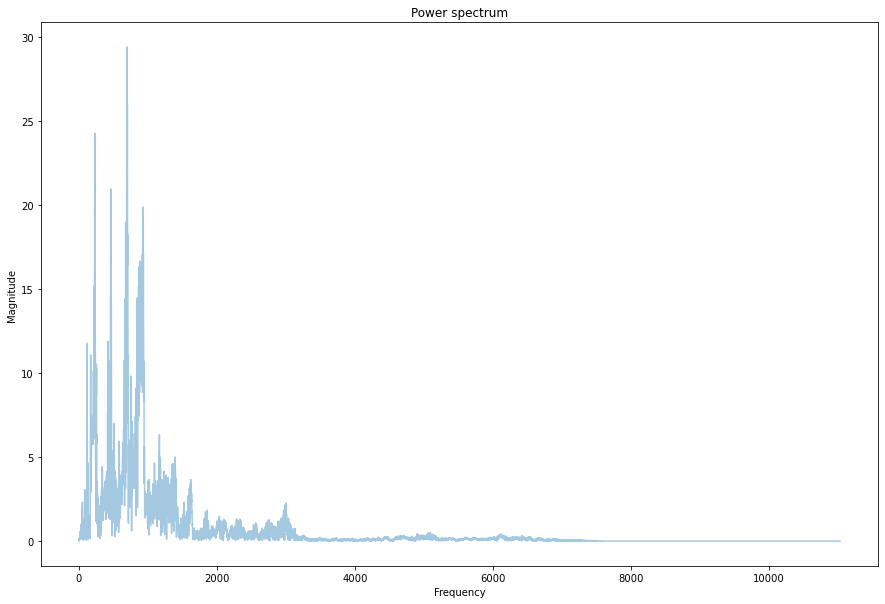
**Bölüm 4. Veri Hazırlama**

Verisetimiz, Kaggle platformunda Google[2] tarafından düzenlenen yarışma için hazırlanmıştır. İngilizce "Yukarı", "Aşağı" gibi tek kelimelik ses komutlarını içermektedir ve ortalama bir saniyelik seslerden oluşur. Yapay Zeka modelimize verilerin eklenmesi için sabit bir boyut gerektiği için, ilk bir saniyelik kısmı dikkate alarak veriler hazırlandı.

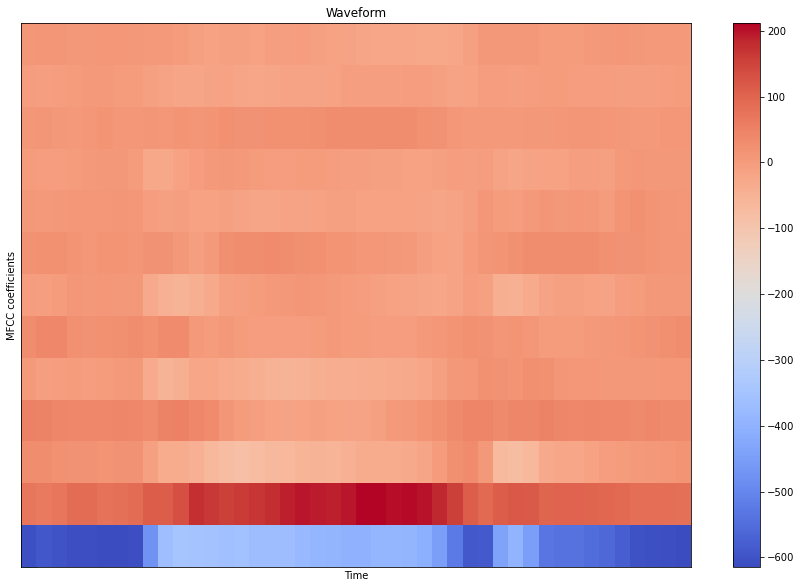
Librosa kütüphanesi ile "Dog" komutunun dalga formu (Şekil 4.1), spektral yoğunluğu (Şekil 4.2) ve MFCC değerleri (Şekil 4.3) çıkartılmış ve görselleştirilmiştir.



Şekil 4.1



Şekil 4.2



Şekil 4.3

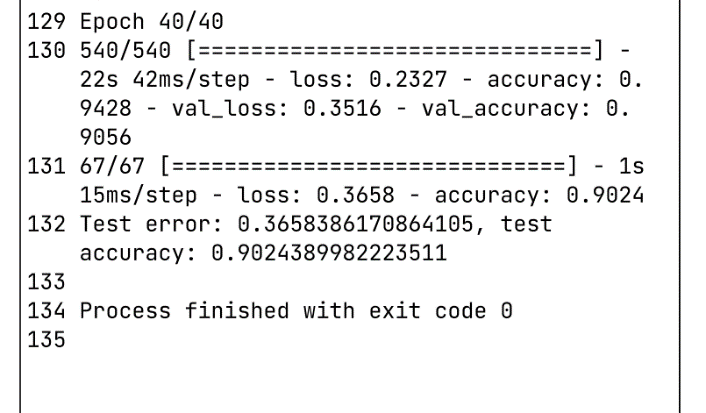
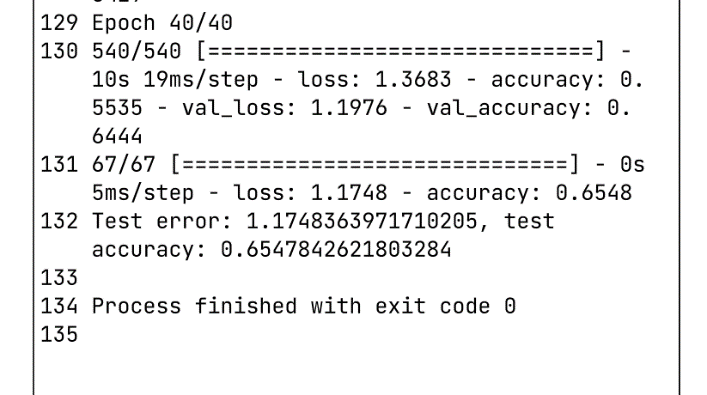
JSON veri yapısındaki veriler, onlara karşılık gelen etiketlerle birlikte hazırlanan dosyada eğitim ve test için tutulmaktadır.

**Bölüm 5. Tasarım**

Model üzerinde hala değişiklik yaptığımız için bütün veri seti yerine sadece 10 sınıfı eğitim ve test aşamalarında kullanmayı tercih ettik. Yalnızca optimizerların genelleştirmeye etkilerini test ederken bütün veri sınıflarını dahil ettik. Projemiz tamamen optimize olduğunda bütün sınıflar ve veriler dahil edilecektir.

Modelimizin tasarımını yaparken sınıflandırma için birkaç çeşit metric, optimizer, denedik. Bunlar literatürde hyperparametre olarak tanımlanmaktadır. İlk başta optimizerı değerini Adam da sabit tutarak epochs değerini 30,40 ve 50 olarak değiştirip sonuçları inceledik. 20 epoch değerinde veri setinin hala öğrenmeye devam ettiğini gördük, 30 epoch değerindeki sonuca baktığımızda test accuracy değerinin max verimi aldığını gördük, 40 epoch değerinde test accuracy değerinin düştüğünü ve artık ezberlemeye girdiğini fark ettik. 50 epoch değerinde ise uç noktada bir ezberleme vardı ve bunun sonucunda yüksek bir train accuracy değeriyle karşılaştık.

Kullanıma açık pek çok optimizer bulunmakla birlikte hepsinin kendi özgü avantaj ve dezavantajları vardır. Biz denemelerimizde Stochastic Gradient Descent (SGD) ve Adam optimizerlarını denedik. Adam, SGD’ye kıyasla daha hızlı çalıştı ancak SGD’nin daha genel verilere hitap ettiğine dair literatür çalışmaları bulduk.

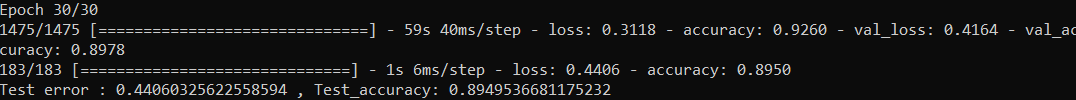
 

Epoch:40, Optimizer: Adam Epoch:40, Optimizer: SGD

Epoch değeri 30, 40 ve 50 olarak Adam ve SGD optimizerlarında test ettik, ilgili sonuçlara kaynakçada verdiğimiz github adresinden ulaşılabilir.[3] Proje henüz geliştirme aşamasında olduğu için yetkilendirme verilmeyen kullanıcılar şu an projemizi görememekteler.

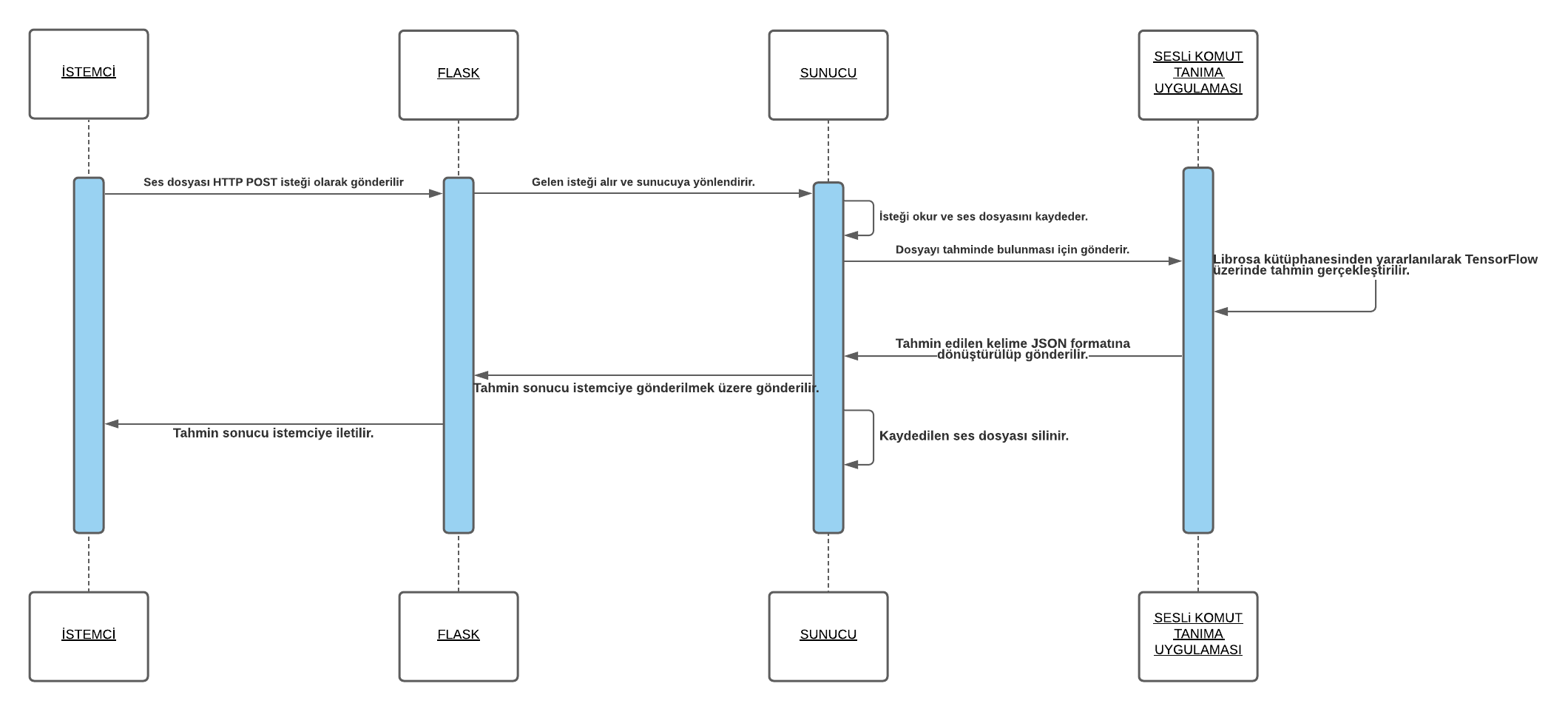
Metric olarak “accuracy” denendi. Sınıflandırma problemi çözmeye çalıştığımız için araştırmalarımız sonucu en iyi sonucu onun verdiğine dayanan çalışmalar gördük. Epoch sayısı ve optimizer çeşidine göre accuracy değerimiz farklılık gösterdi.

Üç katmanlı Convolution işleminden sonra Dense layer ile sınıflandırma yapan modelimizde, dördüncü bir Convolution katmanı eklemek performans kaybına sebep oldu.

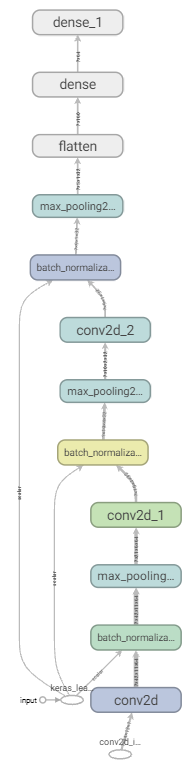


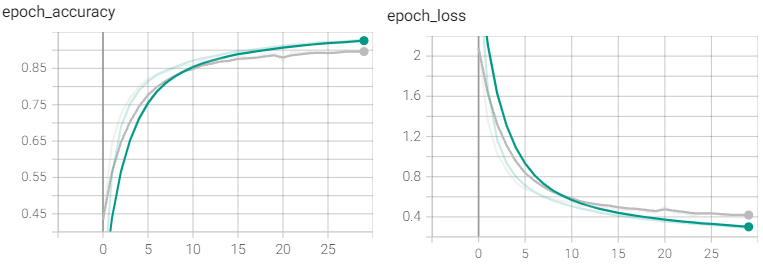
*Dört Convolution katmanlı eğitimin sonucu.*

Sistemin Mimari Tasarımının Sıralama Şeması:



Modelimizi, Tensorflow’a ait TensorBoard modülü ile görselleştirdik.



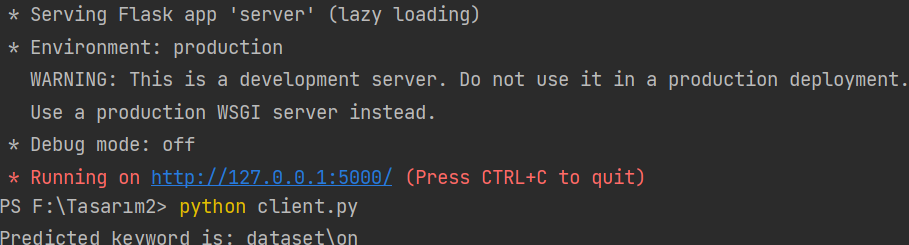


Eğitim ve Validation sırasında loss ve accuracy değerlerindeki değişimin grafiği

**Uygulama**

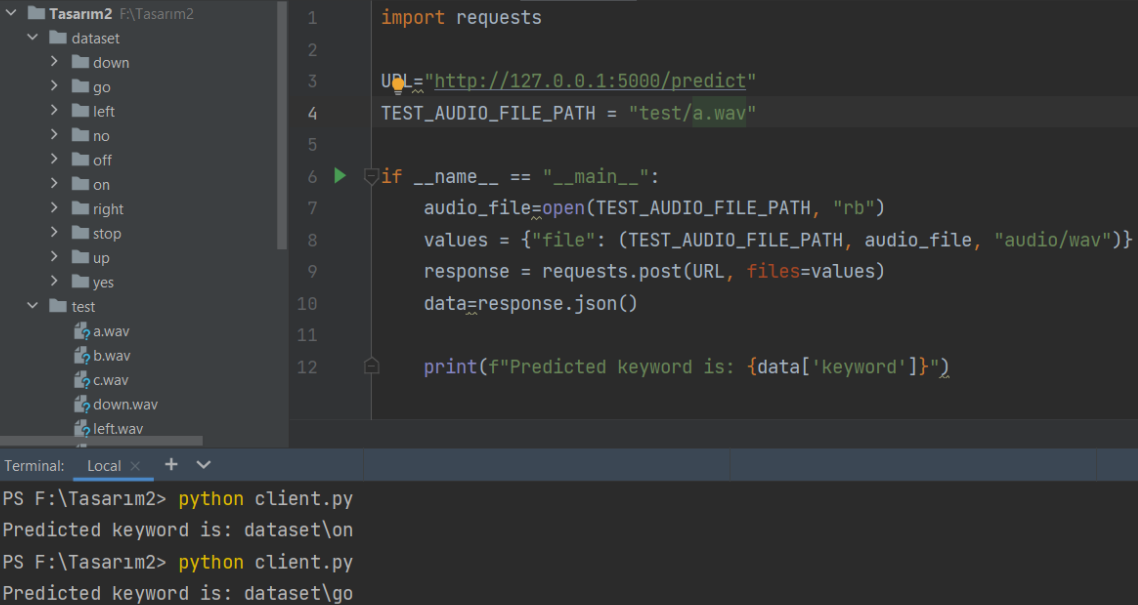
Projemizde bulunan client tarafından gönderilen ses kaydı, 5000 nolu portu dinleyen servera gönderiliyor. Daha sonrasında Singleton tasarım deseniyle hazırlanmış Logger sınıfımızın içerisinde bulunan modelimiz, server tarafında gönderilen ses kaydına bir cevap oluşturuyor. Veri setimizi en başta eğitim ve test olarak ikiye ayırmıştık. Test kısmında bulunan ve hiç eğitimde kullanılmamış verilerimizi bu yolla gönderdiğimizde hep doğru sonuçlar aldık. Deneme amaçlı olarak kendi seslerimizi kayıt altına alarak denemeler de gerçekleştirdik.

Github’ta, kendi oluşturduğumuz seslerden örnekler paylaştık.



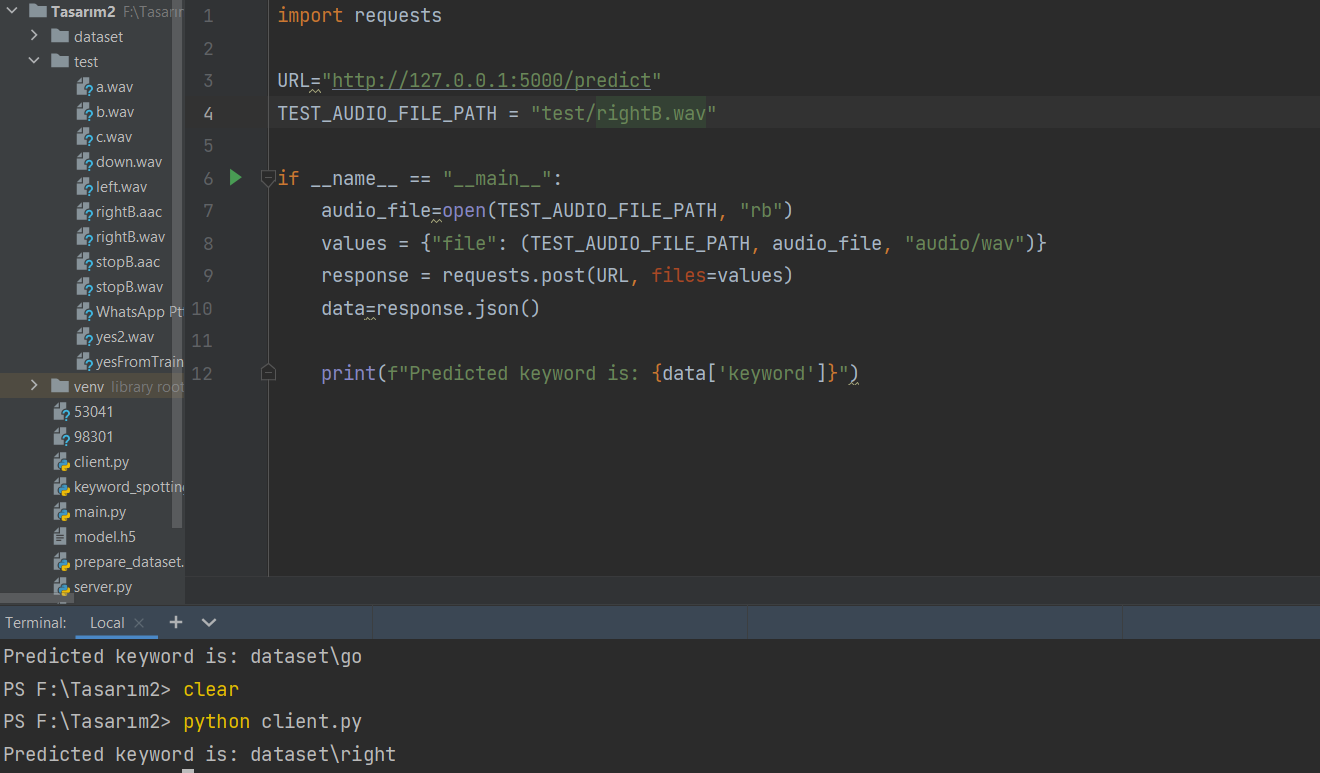
*Client ve server aynı anda çalıştırılarak sonuçlar elde edilmiştir.*

Gürültülü ortamlarda alınan seslerimiz, dışarıdaki seslerin etkisi ve telaffuz farklarından dolayı bazı durumlarda hatalı tespit edildi. Aynı zamanda, bir saniyenin altında oluşan ses kayıtları, boyut farkından dolayı hata durumu oluşturdu. Örneğin, Github repositorymizde bulunan a.wav sesinde, aslında “down” kelimesi telaffuz edilirken, “go” kelimesi tahmininde bulunuldu.



*a.wav ses kaydının sonucu “go” olarak bulunuyor.*

Telaffuzuna dikkat ederek kayıt aldığımız “right” kelimesi ise, sorunsuz bir şekilde tespit edildi.



**Kaynakça**

[1]: Evrişimsel Sinir Ağları**,** Google Tensorflow, Tensorflow Core Öğreticiler

[2]: Sesin Öznitelik Çıkarımı — Librosa, Abdurrahman Oğuzhan Durmaz, Data Runner Veri Bilimi Öğretileri

Voice Command Recognition with Deep Learning, Emre Ates, MASTER OF SCIENCE THESIS

[3]: <https://github.com/muadgra/Tasarim-Projesi>