ELM368 – CIVCIV SESI ILE TAVUK SESI AYRIMI

Efe BAYRAKÇEKEN, Beyzanur CAM, Erhan GÖK
210102002026, 210102002037, 200102002037
e.bayrakceken2021@gtu.edu.tr, b.cam2021@gtu.edu.tr, e.gok2020@gtu.edu.tr

ABSTRACT (ÖZET)

Bu projede 2 tane birbirinden farklı sesin (civciv ile tavuk sesi) ayrımı yapılmıştır. Bunun için pyfda programında çeşitli filtreler tasarlanıp bu filtreler örnek ses sinyallerine göre el ile tasarlanmıştır. Sonra bu filtreler kullanılarak civciv ile tavuk sesini ayırıp, ses sınıflandırılması yapılmıştır. İşlemler Jupyter formatında Python dilinde yazılmıştır.

ANAHTAR KELİMELER
Pyfda, FIR filtre, Ses İşleme, Civciv, Tavuk, Python

1. Giriş

Projenin amacı civciv ve tavuk sesinin tanımlanması ve birbirleri ile iç içe oldukları zaman birbirlerinden ayrılmasıdır.

Projenin temeli, frekans domenindeki civciv ile tavuk sesinin farklılığından dolayı bu sesleri basit birkaç filtre kullanarak ayırabilinmesidir. Bu filtrelerin tasarlanması için ilk başta bu sinyallerin spektrumları çizilip genel olarak civciv sesinin hangi bant aralığında, tavuk sesinin de hangi bant aralığında yer aldığına bakılıp genel olarak birkaç filtre tasarlanmıştır.

Buradaki esas nokta civciv sesi ile tavuk sesi arasında örtüşümün az olmasıdır. Bu özellik, filtreleme işlemlerimizi kolaylaştırmıştır. Bu "basit" filtreler PYFDA programında tasarlanıp, bahsedilen programdan Sabit Katsayılı Fark denkleminin katsayıları çıkartılıp, bu katsayılar kullanılarak filtreler giriş sinyallerine ayrı ayrı uygulanmıştır.

2. Deneyler ve Analiz

Bu problem için 6 tane eğitim verisi seçilmiştir. Bu eğitim verisi internetten bulunmuştur. [i] Bu videolardaki seslerin hepsi bizim gereksinimlerimiz için gereğinden fazla olmasından kaynaklı bütün ses kayıtları 15 saniyelik kesitlere bölünmüştür.

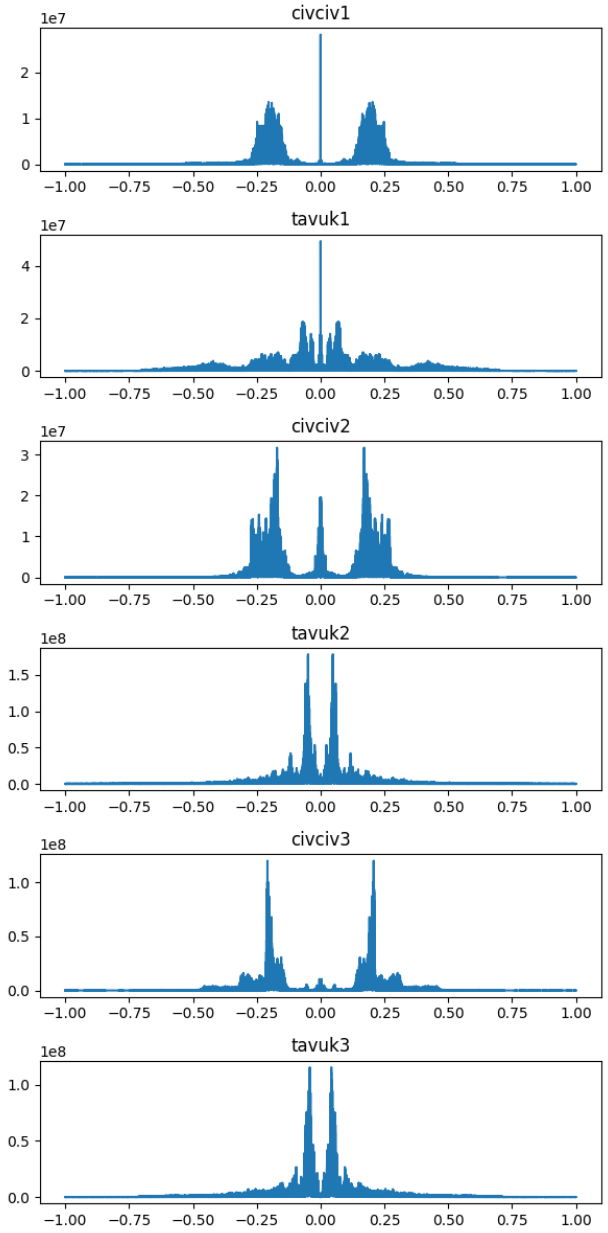
Her şeyden önce gerekli olan kütüphaneler import edilmiştir. Numpy, scipy kütüphaneleri genel olarak sinyal işleme işlemleri için çok büyük kolaylıklar sunarken matplotlib kütüphanesi pythonda grafik çizimini sağlar. Scipy.io ise, wav dosyalarının okunulması, filtre dosyalarının okunulması gibi önemli işlemlerde işimize yaramıştır. Ipython kütüphanesi ise jupyter ortamında ses dosyaları oynatma gibi basitlikler sağlar.

Öncelikle bu ses sinyalleri orijinal halleri ile gösterilmiştir. Bu ses sinyalleri beklendiği gibi değişmesizdir.

Frekans domeninde analiz yapmak için bu sinyallerin Fourier dönüşümleri alınmıştır.

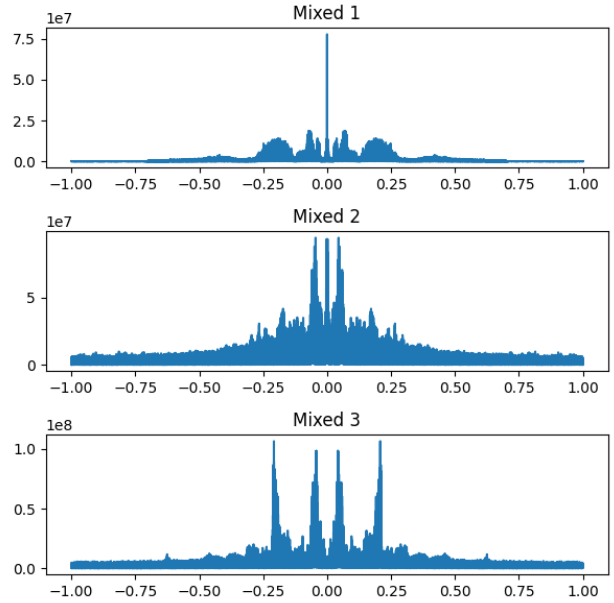
Frekans domeninde, giriş kısmında iddia edilen "civciv sesi ile tavuk sesi arasında örtüşümün az olması" iddiasının doğruluğu belli olmaktadır. Civciv sesi büyük oranda 0.119pi frekansı üstünde, tavuk sesi ise 0.103pi frekansı altında çıkmıştır.

Bu ses sinyallerine ekteki dosyalardan dinlenilebilir.



Şekil 1 Test verilerinin frekans analizi.

Bir sonraki adımda bu gösterilen sinyaller bir tavuk, bir tane de civciv olarak eşleştirilip birleştirilmiştir. Bu "karıştırılmış" sinyaller ileriki adımlarda filtrelenme işlemleri için kullanılacaktır. Bu sinyallerin de spektrumları çizilmiştir. Bu sinyaller hem tavuk hem de civciv sesi karakterlerini de göstermiştir.

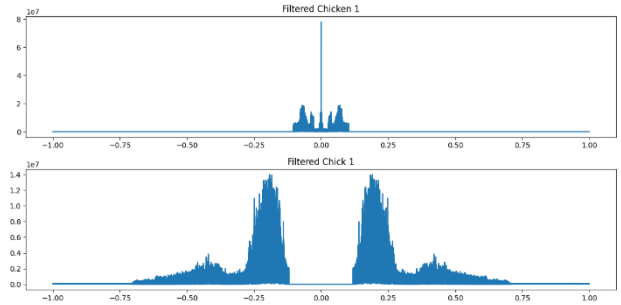


Şekil 2 Karıştırılmış sinyallerin frekans analizi.

Bu sinyalde, bahsedilen bütün bantlarda içeriği vardır.

Örnek olarak bu sinyaller frekans domeninde ideal filtrelerden geçirilip, zaman domenine geri dönülmüştür. Bu işlemler yapıldıktan sonra spektrumlar bir daha çizilmiştir. Beklenildiği üzere bu filtreler çok keskin kesimler yapmıştır. Ancak bu filtreler gerçeklenemez. Bu filtre, frekans domeninde ideal bir dikdörtgen dalgadır ve bu filtrenin zaman domeninde eşdeğeri sinc dalgasıdır. Sinc dalgası -sonsuzdan +sonsuza giden bir dalga olmasından kaynaklı bu filtre nedensellik özelliği göstermez. Bu da gelecek ile alakalı önceden bilgiye ihtiyacımızın olduğunu gösterir. Tabiki de böyle bir şey imkansızdır.

Örnek olarak 1. karma sinyalin filtrelenmiş hali aşağıda çizdirilmiştir. Bu sinyal yukarıda da bahsedildiği gibi ideal filtre ile kesilmiştir. Tavuk filtresi 0.103pi bant genişliği olan bir filtredir. Civciv filtresi ise 0.119pi üstünü geçiren bir üst geçiren filtredir.



Şekil 3 İdeal Filtre kullanılarak filtrelenen ses.

Bu filtrelerden geçen bu karma sinyaller rapor ile birlikte teslim edilen jupyter dosyasından dinlenilebilir.

Bu filreler uygulanırken numpy kütüphanesinin np.where komutu kullanılmıştır.

```
filtered_chick_w = np.where(abs(w)>0.119*pi, mixed_1_W, 0)
filtered_chicken_w = np.where(abs(w)<0.103*pi, mixed_1_W, 0)
```

Bu komut ile w'nın belli bir sayıdan büyük olduğu yerler alınmıştır.

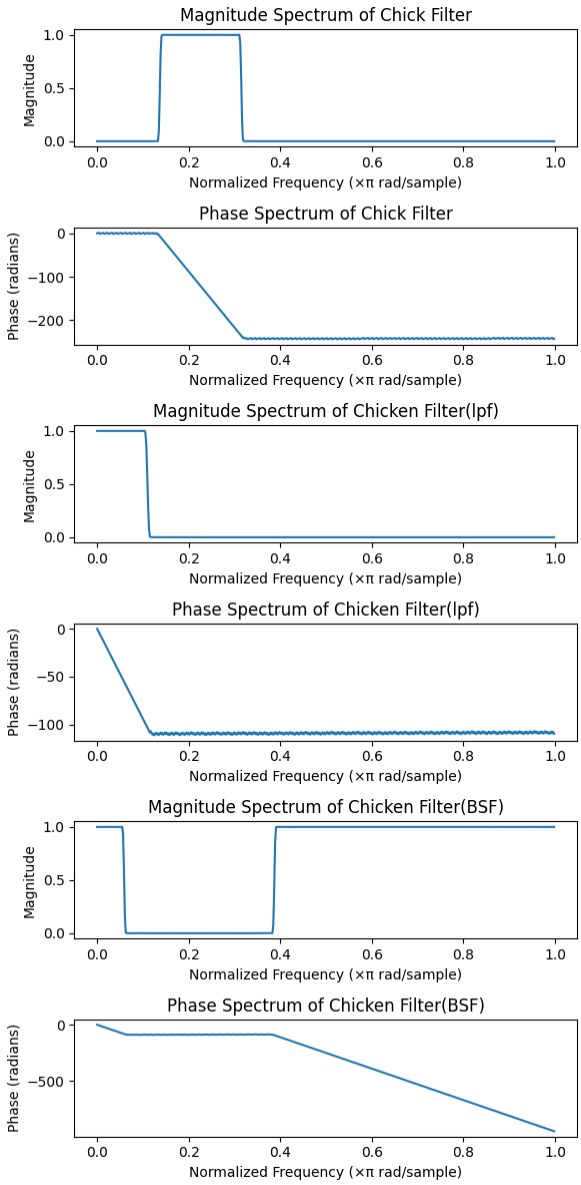
Filtre tasarlarken belirtildiği gibi pyfda kullanılmıştır. Bu filtreler tasarlanırken ideal filtreler yerine ideal olmayan ancak diğer bir sürü konuda destek sağlayan sonlu dürtü cevaplı filtreler kullanılmıştır.

Tavuk sesinin filtresi için "Windowed fir" filtre kullanılıp bu filtrenin çeşidi olarak da keiser window seçilmiştir.

Civciv sesi ise de aynı "Windowed fir" filtre kullanılmıştır ancak bu filtrede diğerinden farklı olarak keiser window yerine daha popüler olan hamming window kullanılmıştır. Hamming window diğer frekanslardan karışımı azaltmakta iyi olması bakımından avantajları vardır ancak keiser window ise çok daha esnek bir pencere çeşididir.

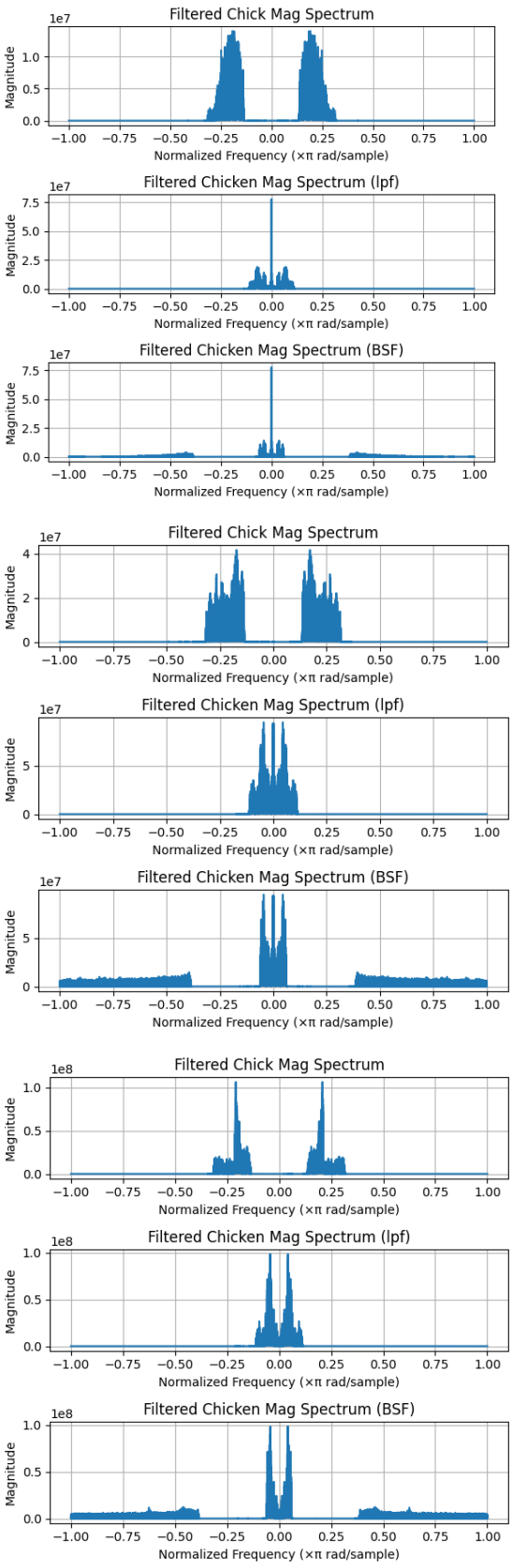
Bu filtreler tasarlandıktan sonra standart .mat dosyası olarak PYFDA programından pythona aktarılmaya uygun bir dosya olarak kaydedilebilir.

Biraz daha idealden farklı olarak civciv filtresi pyfda'da band pass filtresi kullanılarak yapılmıştır.



Şekil 4 Kullanılan tasarlanmış filtreler.

Burada tavuk sesi filtresi için 2 adet filtre tasarlanmıştır. Birisi, sadece basit bir alçak geçiren filtresi olup diğeri ise bant durduran filtresidir. Bu filtre civciv frekans aralığındaki sesleri azaltır. İki yaklaşımın da kendilerine has avantajları vardır. Saf alçak geçiren filtre yüksek frekanslardaki çıkabilecek karışmaları engellemekte daha iyi olur ancak tavuk sesi dışında bir sürü sesi de filtreler. Bu da sesin daha boğuk çıkmasına neden olur.



Şekil 5 Filtrelenmiş seslerin genlik spektrumları.

Yukarıda bahsedilen bütün ses sinyalleri için bahsedilen bütün filtreler uygulandıktan sonra spektrumlar gösterilmiştir. Bütün bu işlemler sonrasında sinyaller dinlenince tavuk filtresi uygulanan sinyallerde sadece tavuk sesi, civciv sesi uygulanan filtrelerde ise sadece civciv sesi gelmiştir. Filtreleme işlemi başarılı olmuştur.

Ancak bu kanı tamamen de doğru değildir. Bu filtrelemeler gerçekleştirildiğinde ses biraz da olsa kalitesini kaybetmiştir.

Bunun sebebi ise tavuk ve civciv seslerinin net kesim frekanslarının içinde aslında kalmayıp, seslerin harmonikleri yüzünden o birbirlerinin frekanslarının içerisine karışmasıdır. Ayrıca Tavuklar ve Civcivler tek bir ses yapmazlar. Tavuklar birkaç farklı koşul için farklı birkaç adet ses çıkarır. Civcivler de annesi ve kardeşleri arasında konuşurken farklı ses çıkarırlar. Bu net ayrım yapılmasını zorlaştırır.

Bahsedilen filtreler uygulanırken pythonda bir filtre "Class"'ı belirlenmiştir. Bu sınıf genel olarak bizim istediğimiz filtre ile ilgili genel birkaç işlemi yapmamızı kolaylaştırır. Örnek olarak bu filtreleme işlemlerinde her bir filtre çeşidi için yeni bir filtre objesi oluşturulup, bu filtre objesinin methodları kullanılmıştır.

Bu sınıf objelerde ilk oluşturulduğunda Sabit Katsayılı Fark denklemlerinde gerekli olan a, b katsayıları ile başlatılır. Sonraki adımlarda bu katsayılar kullanılarak filtreleme işlemleri yapılır.
Bu sinyalin tavuk sesi mi civciv sesi mi olduğunu ayırt etmekte bu sinyaller civciv ve tavuk filtresinden ayrı ayrı geçirilmiştir. Bu filtrelerden geçirdikten sonra filtre çıkışındaki sinyallerin gücüne bakılmıştır. Eğer tavuk filtresi çıkısı daha güçlü bir çıkış vermiş ise algoritma bu sesi tavuk sesi olarak tanımlayıp tavuk fotoğrafı gösterir. Eğer civciv sesi ise civciv fotoğrafı gösterir.

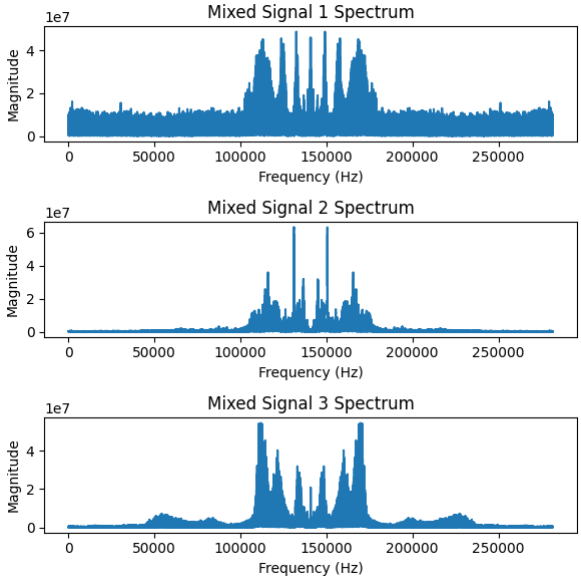
Şekil 6 Is it chick or chicken? çıktısı.

Ancak bu yukarıda yaptıklarımızın hepsi test verisi üzerindendi. Bu filtreler bütün tavuk sesine uyumluluğu bir sonraki adımlarda test edilmiştir.

Bunun için 6 adet daha test sesi kullanılmıştır.

Test ses sinyalleri öncelikle python'a aktarılıp spektrumları çizilmiştir. Bu grafiklere tek başına bakıldığında bile filtrelerin çalışması hakkında yorum yapılabilir.

Eğitim ve test verimizdeki tepeler ve çıkıntılar aynı frekanslarda görüntülenmektedir.



Şekil 7 Karıştırılmış Sinyallerin Spektrumları.

Bu test sesleri de aynı filtredelerken geçirilmiştir.

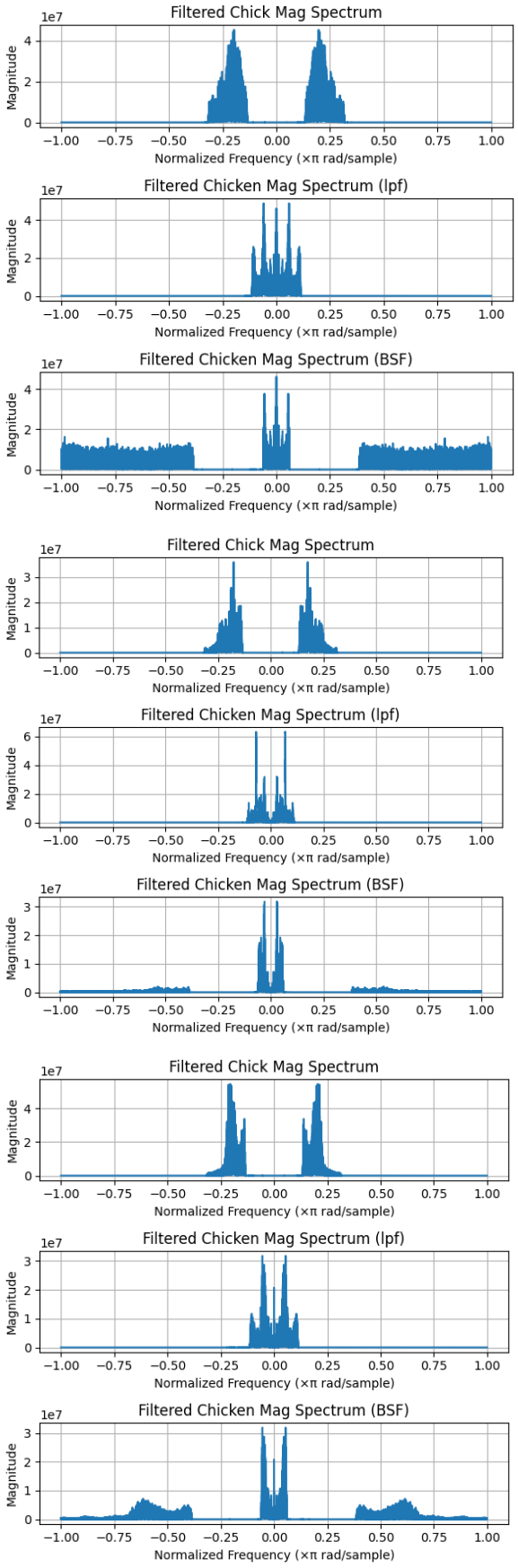
Filtrelerden geçirdikten sonra tavuk ve civciv sesi önceki bölümlerdeki gibi ayrılabilmiştir.

Bunun sebebi de önceden bahsedildiği gibi tavuk ve civciv sesinin tavuktan tavuğa ve civciv sesinin civcivden civcive çok da farklı olmamasıdır. Bu filtreleme işlemlerini kolaylaştırmıştır.

Aşağıdaki grafiklerde farklı çeşitlerde filtrelenmiş sinyaller gösterilmiştir.

Band-stop filtre kullanılan tavuk sesi daha net gelmesine rağmen filtreleme daha az efektif olmuştur. Seste hala civciv sesi belli olmaktadır. Alçak geçiren filtreden geçirilen tavuk sesinde ise civciv sesi yok denilecek kadar azdır.

Civciv sesleri kendi frekans bantlarında yeteri kadar dominant olduğu için civciv sesinde böyle küçük problemler yaşanmamıştır.



Şekil 8 Filtrelenmiş Sesler ve Spektrumları.

Bu filtreler test sinyalleri dışında da efektif olmuştur. Bu filtrelerin genel kullanılabilirliğini göstermiştir.

3. Sonuç ve Yorum

Bu projede civciv ve tavuk sesi olmak üzere iki ayrı sesin ayrımı başarıyla sağlandı. Pyfda programında tasarlanan çeşitli filtreler örnek ses sinyallerine göre manuel olarak tasarlanarak civciv ve tavuk seslerinin sınıflandırılması gerçekleştirildi. Tüm süreç Python'da Jupyter formatı kullanılarak uygulandı.

Bu çalışma, civciv ve tavuk seslerini, frekans domenindeki farklılıklarına dayalı olarak basit filtreler kullanarak ayırmanın mümkün olduğunu göstermiştir. Pyfda'da tasarlanan FIR filtreleri kullanılarak elde edilen sonuçlar, ses sinyallerinin belirli frekans bantlarına göre filtrelenmesiyle etkili ses ayrımının sağlanabileceğini gösterdi.

Bu proje ses sınıflandırması ve sinyal işleme konusunda önemli bir deneyim kazandırdı. Önerilen çözüm, frekans bantlarına dayalı ses ayrımının, belirli sınırlamalara rağmen mümkün ve etkili olduğunu gösterdi. Ancak filtreleme işlemi sırasında ses kalitesinde bir miktar kayıp gözlemlendi. Bu kaybın nedeni civciv ve tavuk ses harmoniklerinin birbirlerinin frekans aralıklarında örtüşmesi ve seslerin keskin kesme frekanslarına sahip olmamasıdır.

Çözümü geliştirmek için daha gelişmiş filtreleme teknikleri ve uyarlanabilir filtreler kullanılabilir. Bu yaklaşım, daha karmaşık ses karışımlarının ayrılmasında daha yüksek doğruluk sağlayabilir. Ek olarak, daha geniş bir veri kümesinin kullanılması, filtrelerin farklı civciv ve tavuk seslerindeki genel performansını artırabilir.

Bu çalışma, ses işleme ve sınıflandırma konusunda pratik bilgi ve deneyim sunarak gelecekteki daha kapsamlı araştırmalara temel teşkil etmiştir. Filtre tasarımı ve uygulama sürecinde karşılaşılan zorluklar ve elde edilen bulgular, bu alanda daha etkin çözümlerin geliştirilmesine katkı sağlamaktadır.

Kaynaklar

[1] DSP Proje Playlist. [Online]. Available: https://youtube.com/playlist?list=PLDAI7S9MCUTa_Nv3oAZDtIincRvtIX1J5&si=xE-WgfOaiu40sRi7. [Accessed: 22-May-2024].

[2] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[3] pyfda ve filtre tasarımı 2, by Köksal Hocaoğlu, mp4. Provided by the course provider.

[4] "ELM368 Ödev-6 (2023-2024 Bahar)," by Köksal Hocaoğlu, PDF file, Provided by the course provider.

[i] İlgili videolar: https://www.youtube.com/playlist?list=PLDAI7S9MCUTa_Nv3oAZDtIincRvtIX1J5