KTO KARATAY
ÜNİVERSİTESİ

T.C.
KTO Karatay Üniversitesi
Fen Bilimleri Enstitüsü

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

HAYVAN SESLERİNİ AYRIŞTIRMAK İÇİN MODİFİYE EDİLMİŞ BİR OTOKODLAYICI MİMARİSİ YAKLAŞIMI

Ahmet Sinan ÖZBAYGIN

KONYA

Kasım 2019

HAYVAN SESLERİNİ AYRIŞTIRMAK İÇİN BİR OTOKODLAYICI YAKLAŞIMI

Ahmet Sinan ÖZBAYGIN

KTO Karatay Üniversitesi Fen Bilimleri Enstitüsü

Elektrik ve Bilgisayar Mühendisliği Ana Bilim Dalı
Yüksek Lisans Programı

Yüksek Lisans Tezi

Kasım, 2019

KABUL VE ONAY

Öğrenci Ahmet Sinan ÖZBAYGIN tarafından hazırlanan "Hayvan Seslerini Ayrıştırmak İçin Modifiye Edilmiş Bir Otokodlayıcı Mimarisi Yaklaşımı" başlıklı bu çalışma, 19 Kasım 2019 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Başkanı: **Dr. Öğr. Üyesi Ali O. ÇIBIKDİKEN**
Necmettin Erbakan Üniversitesi

Tez Danışmanı: **Dr. Öğr. Üyesi H. Oktay ALTUN**
KTO Karatay Üniversitesi

Jüri Üyesi: **Dr. Öğr. Üyesi Ali ÖZTÜRK**
KTO Karatay Üniversitesi

Jüri tarafından kabul edilen bu çalışmanın Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

Prof. Dr. Hüseyin Bekir YILDIZ
Enstitü Müdürü

BİLDİRİM

Enstitü tarafından onaylanan Yüksek Lisans tezimin tamamını veya herhangi bir kısmını basılı veya dijital biçimde arşivleme ve aşağıda belirtilen koşullar dahilinde erişime açma iznini KTO Karatay Üniversitesine verdiğimi bildiririm. Bu izinle, Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak ve gelecekteki çalışmalar (makale, kitap, lisans, patent vb.) için tezimin tamamının veya bir bölümünün kullanım hakları yalnızca bana ait olacaktır.

Tezimin bütünüyle kendi çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Telif hakkı bulunan ve sahiplerinden yazılı izinle kullanılması zorunlu olan kaynakları, yazılı izin alarak kullandığımı ve istenildiğinde izinlerin suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayımlanan "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" kapsamında, tezim, aşağıda belirtilen koşullar haricince, YÖK Ulusal Tez Merkezi ve KTO Karatay Üniversitesi Açık Erişim Sisteminde erişime açılır.

☐ Enstitü / Fakülte Yönetim Kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.[1]

☐ Enstitü / Fakülte Yönetim Kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir.[2]

☐ Tezimle ilgili gizlilik kararı verilmiştir.[3][4]

19 Kasım 2019

Ahmet Sinan ÖZBAYGIN

[1] MADDE 6(1) Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.

[2] MADDE 6(2) Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.

[3] MADDE 7(1) Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler YükseköğretimKuruluna bildirilir.

[4] MADDE 7(2) Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.

ETİK BEYAN

KTO Karatay Üniversitesi Fen Bilimleri Enstitüsü Tez Hazırlama ve Yazım Kurallarına uygun olarak Dr. Öğr. Üyesi H. Oktay ALTUN danışmanlığında tarafımdan üretilen bu tez çalışmasında; sunduğum tüm veri, enformasyon, bilgi ve belgeleri bilimsel etik kuralları çerçevesinde elde ettiğimi, tüm değerlendirme, analiz, bulgu ve sonuçları bilimsel usullere uygun olarak sunduğumu, tez çalışmasında yararlandığım kaynakların tümüne bilimsel normlara uygun biçimde atıfta bulunarak kaynak gösterdiğimi, tezimin kaynak gösterilen durumlar dışında özgün olduğunu bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

19 Kasım 2019

Ahmet Sinan ÖZBAYGIN

Özet

HAYVAN SESLERİNİ AYRIŞTIRMAK İÇİN MODİFİYE EDİLMİŞ BİR OTOKODLAYICI MİMARİSİ YAKLAŞIMI

Ahmet Sinan ÖZBAYGIN

KTO Karatay Üniversitesi,
Fen Bilimleri Enstitüsü,
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi H. Oktay ALTUN

Kasım 2019

Hayvan davranış bilimi, hayvanların davranışlarını anlamak suretiyle, onları kontrollü ortamlarda tutmak ve evcilleştirmek gibi konularda bize yardımcı olurken, aynı zamanda hayvan türlerini ve canlılığı daha iyi anlamamızı da sağlar. Bu alanlarda yapılan araştırmalarda veriler doğa ortamından toplanabilirken, bazen de kontrollü ortamlarda tutulan hayvanlardan faydalanılmaktadır. Bu alandaki araştırmalarda veri toplama sırasında hayvan sesleri karışık şekilde kaydedildiğinde, bunları ayırarak analiz etmek gerekmektedir. Hayvan sesleriyle oluşan kokteyl partisi problemi için derin sinir ağlarından ve derin öğrenme metodolojisinden faydalanarak yeni bir çözüm önerisi sunmaktayız. Bu tezde derin öğrenmede sıkça kullanılan otokodlayıcı mimarisini değiştirerek yeni bir derin öğrenme mimarisi oluşturduk. Bu mimariyi kullanarak hayvan seslerini birbirinden ayırdık. Getirdiğimiz yakaşımın performansını değerlendirdik.

Anahtar kelimeler: Derin öğrenme, makine öğrenmesi, yapay sinir ağı, kaynak ayrıştırma, kokteyl partisi problemi, otokodlayıcı mimarisi

Abstract

A MODIFIED AUTO-ENCODER ARCHITECTURE APPROACH FOR SEPARATING ANIMAL SOUNDS

Ahmet Sinan ÖZBAYGIN

KTO Karatay University,
Institute of Science,
Master of Science Thesis in Electrical and Computer Engineering

Advisor: Asst. Prof. H. Oktay ALTUN

November 2019

Animal behavioural science aids us to understand animal species and their life styles, while helping us in tasks like keeping them in controlled environments or taming them. While data can be collected from natural environment on researches at these fields, sometimes animals that are being held at controlled environments can be benefited from. For researches at these fields related to animal behavioural science, on many occasions, animal sounds are being recorded as mixture of animal and environmental sounds while collecting data. These sounds need to be first separated and then analyzed. For this cocktail party problem consisting animal sounds, we present a new solution proposal that benefits from deep neural networks and deep learning methodology. In this paper, we created a new deep learning architecture by modifying auto-encoder architecture, being widely used in deep learning research. Using this architecture we separated animal sounds from each other. We evaluated performance of the approach we developed.

Keywords: Deep learning, machine learning, artificial neural network, source seperation, cocktail problem, autoencoder architecture

Teşekkür

Çalışmalarım boyunca her türlü fedakarlığı gösteren ve bana sürekli destek olan, moral ve motivasyon konusunda beni hiç yalnız bırakmayan bütün aileme, birebir çalışarak fikirleriyle projeyi yöneten ve katkılarıyla beni yönlendiren tez danışmanım Dr. Öğr. Üyesi Hüseyin Oktay Altun'a, çalışmamdaki desteklerinden ötürü Cihan Çalışır'a teşekkür ederim.

Ahmet Sinan ÖZBAYGIN
Kasım-2019

İçindekiler

Şekil Listesi

Tablo Listesi

Simgeler

Simgeler	Açıklama
$\mathbf{h}(x)$	Aktivasyon fonksiyonu
y_i	Orjinal dizinin i indeksi
dB	Desibel

Kısaltmalar

Kısaltmalar	Açıklama
$DARPA$	Defense Advanced Research Projects Agency
$EASI$	Equivariant Adaptive Separation via Independence
ICA	Independent Component Analysis
EEG	Electroencephalographic
$TF-UBSS$	Time-Frequency Underdetermined Blind Source Separation
$BASS$	Blind Audio Source Separation
ASR	Automatic Speech Recognition
PC	Personal Computer
GPU	Graphics Processing Unit
WAV	Waveform Audio File
DCT	Discrete Cosine Transform
FNN	Feedforward Neural Network
$FCNN$	Fully-Connected Neural Network
$FFCNN$	Feedforward Fully-Connected Neural Network
$RELU$	Rectified Linear Unit
$SELU$	Scaled Exponential Linear Unit
$LPCM$	Linear Pulse Coding Modulation
MSE	Mean Squared Error
$PSNR$	Peak Signal-to-Noise Ratio

1 Giriş

Hayvan davranış bilimi, hayvanların davranışları özelinde, karakteristiklerini, eğilimlerini, yönelimlerini, rutinlerini, düşünce ve yaklaşımlarını inceleyen uygulamalı bir bilimdir. Bu bilim bizim hayvanların davranışsal örüntülerini öğrenmemizi sağlar. Onları anlamamıza ve gerektiğinde onları korumak için kontrollü ortamlarda tutmak veya iş gücü olarak faydalanmak için evcilleştirmek gibi konularda bize yardımcı olur. Aynı zamanda hayvan türlerini ve canlılığın davranışsal temellerini daha iyi anlamamızı sağlar. Bu alanda çalışan bilimciler günümüzde çiftlik, hayvanat bahçesi, laboratuvar, korunmuş vahşi yaşam alanları gibi kontrollü ortamlarda tutulan hayvanlardan faydalanmaktadırlar. Çünkü bu ortamlarda o türe ait canlıları gözlemlemek çevrenin kontrol altında olması sayesinde hem daha güvenlidir, hem türe ait toplulukların, kendisi için yaşam şartları sağlanmış belirli bölgelerde gözlenmesi daha kolaydır. Bu pratiği doğada eylemleştirmeyi zorlaştıran temel unsurlardan bir tanesi, çok sayıda farklı türden hayvanların, bitkilerin ve cansız unsurların ürettiği çevre seslerinin doğal ortamda birbirine karışmış olarak işitilmesidir. Halbuki ilgilenilen türe ait davranışsal bilgiler toplanırken, sadece o türe ait seslerin net olarak dinlenilebilmesi gerekmektedir. Bu noktada kokteyl partisi problemi, bir kaynak ayrıştırma problemi olarak karşımıza çıkar.

Kaynak ayrıştırma problemi, birbirinden bağımsız kaynaklardan (veri kümesi) oluşan karışımın, karışımı oluşturan kaynaklara ayrılmasını amaçlayan bir problemdir. Sesten, çok katmanlı resimlere, sismik verilerden, elektromanyetik sinyallere kadar verinin birden fazla doğrusal bağımsız kaynak içerdiği her problem ve çözüm uygulaması, bu başlık altında toplanır. Buna örnek olarak kokteyl partisi problemi verilebilir.

Doğal işitsel ortamlar, ister doğal ortamlar olsun, ister şehir ortamı olsun, aynı anda ses üreten çok sayıda unsur (kaynak) içerirler. Kokteyl parti problemi, ilgilenilen sesi bu tür karmaşık işitsel ortamlar içerisinde takip etme görevidir. İsmini aldığı senaryo olan kokteyl partisi senaryosunda da çok sayıda problem örneği mevcuttur. Dinleyici eşzamanlı olarak ses üreten çok sayıda ses kaynağından gelen sesleri işitirken, ilgilendiği bir konuşmayı takip etmelidir. Doğal olarak zor bir prob-

lemdir ve insanların bunu nasıl çözdüğü konusuna karşı uzun zamandır süregelen bir ilgi vardır. Bu problem ile etkileşmek insanlara has bir durum değildir. Tıpkı davranış, vücut dili ve hormonlarda olduğu gibi hayvanlar da iletişim ve etkileşim için sesi kullanmaktadırlar. Sürü halinde yaşayan türler, kendi oluşturdukları "kokteyl partisi"nde birbirleriyle iletişim kurabilmek için, bu problemi çözmek zorundadırlar.

Kokteyl partisi probleminde kavramsal olarak ayrık iki ana unsur vardır. Birincisi, ses ayrıştırma problemidir. Bu problem tabanında inceleyecek olursak, işitsel ortamdaki ses, ortamdaki tüm kaynaklarca üretilen seslerin toplanmasıyla oluşur. Toplanma neticesinde oluşan ses karışımı, dinleyicinin kulağına gider. Dinleyici karışımda ilgilendiği ses kaynağı dışındaki hiçbir kaynağın ürettiği sese ilgi duymaz. Fakat ilgi duyduğu sesi, ilgi duymadığı seslerle oluşturduğu karışımdan ayırabilmek için, sadece duyduğu karışık sese dayanarak, ses karışımındaki her bir kaynağa ait sesleri, zihninde ayrı ayrı canlandırabilmelidir. Ancak bu yapıldıktan sonra ayrıştırma mümkün olabilir.

Ana unsurlardan ikincisi ise, dinleyicinin zihnen, ilgilendiği kaynak dışındaki kaynakları ihmal ederken, ilgilendiği kaynağı takip etmesidir. Algı süreçlerimizin çoğu, aynı anda sadece bir hedef üzerine yoğunlaşabilir. Bu sebeple problemin bu unsuru esasen ses ayrıştırma ile iç içedir. Beraber çözülme gerekliliği vardır ve ayrıştırma unsurunda bahsedildiği üzere, farklı kaynakların karakterize edilebilmesi için, dikkatin kaynaklar arası geçişi ve birden çok kaynağın, dinleyici için en fazla anlam ifade edecek şekilde zamanlanarak, kesik parçalar halinde takip edilmesi gerekmektedir.

Kokteyl partisi Cherry'nin 1953'teki makalesi ile popülerleşmiştir. Cherry, problemin dikkat kısmına yoğunlaşmıştır. Gözlemcilerin bir konuşma sinyalini diğerinin üzerinde bir öncelikle seçip seçemeyeceği, seçmedikleri sinyal ile alakalı bir bilgi elde edip edemedikleri, ve dikkatlerini nasıl sinyaller arasında değiştirebildikleri üzerine çalışmak maksadıyla, bugün yaygın olarak bilinen ve kabul görmüş, dikotik dinleme paradigmasını ortaya atmıştır. Dikotik dinlemede sol ve sağ kulağa, birbirinden farklı birer ses kaynağı aynı anda dinletilir [1].

Yirmi yıl sonra Bregman ses ayrıştırma üzerine çalışmaya başladı. Çalışmasını ise işitsel sahne analizi ismiyle terimleştirdi. Kokteyl partisi problemi üzerine günümüzde yapılan çalışmaların çoğu, ayrıştırma problemi ana unsuru üzerine temellendirilmiştir. Bunun sebebi ise, zihinsel olarak dinlemeye dönük sergilenen işitsel seçicilik mekanizması, bu mekanizmanın ayrıştırma problemi ile arasındaki bağlantı ve nedensellik hakkında çok daha az bilgimizin olmasıdır.

Kokteyl partisi probleminin zorluğu, ses ayrıştırma probleminin zorluğu ile yakından ilgilidir. Zor bir algısal problemdir. Fiziksel ortamda işitilen bir ses

karışımının içerdiği birçok ses kaynağına ait sesin örüntüleri, fiziksel olarak karışım ile tutarlıdır. Gerçek olan tek ses de karışımın kendisi olduğundan, ilgilenilen kaynaklara ait seslerin daha isabetli ayrılabilmesi adına, bu ses kaynaklarına ait, karışım içinde fiziksel tutarlılık bağlamında gizlenmiş olan öznetliklerin verilen karışımdan çıkarılabilmesi gerekmektedir. Zihnimiz bu konuda çoğunlukla iyi iş çıkarır.

Problemin çözümlenmesi sürecinde, öncelikle kulaktan işitilen sesin, işitsel merkezde frekans ayrıştırması yapılmaya çalışılır. Bu adımdan itibaren problemin çözümlenmesi ile ilgili iki temel problem ortaya çıkar. Birincisi, ses karışımını oluşturan ses kaynaklarının, ses karışımı ile fiziksel tutarlılıklarından ileri gelen, karışım içindeki hangi parçanın hangi sese ait olduğunun net olarark bilinememesidir. Karışımda ilgilenilen kaynağa ait enerji düşükken karışıma ait enerjinin yoğun olduğu birçok nokta vardır. Bu noktalarda, karışım içinde ilgilenilen kaynak haricindeki kaynakların oluşturduğu karışımın ses karakterinin çözümlenerek, karışım içinde ilgilenilen kaynaktan ayrıştırılması gerekliliği hasıl olur. Çözümleme adına benzer bir rutin üzerindeki bir diğer senaryo, ikinci temel problemi ortaya koyar. Karışımın belli noktalarında, ilgilenilen kaynağın enerjisi yoğun iken, karışımın enerjisi daha yoğun olur, bu da ilgilenilen kaynağın fiziksel olarak karışım tarafından maskelenmesi anlamına gelir. Yani sesin bu noktasından ilgilendiğimiz kaynağa dair bilgi alma olanağımız fiziksel olarak engellenmiştir.

Dolayısıyla kokteyl partisi ve benzeri işitsel ortamlardaki ses ayrıştırmaları aşırı derecede zor problemlerdir. Tipik örneklerde dinleyicinin tüm dikkatine ihtiyaç vardır. İhityaç duyulan konsantrasyon anlık olarak, dinleyicinin işitme haricindeki temel algısal hassasiyetlerini dahi minimuma indirebilir. Bunlardan ayrı olarak karışımı oluşturan kaynak sayısı, kaynak konumları (sesin geliş açısı ve mesafesi), konumsal olarak yakın ve uzak olan kaynakların sayısı, takip edilen kaynak sayısı ve bu kaynaklara ait parametreler (bu parametrelerin benzerlikleri ve farklılıkları) problemin karmaşıklık derecesine, dolayısıyla zorluk derecesine etki eden diğer temel unsurlardır [2].

Bu problemin çözümünün sağlayacağı, kaynak karışımları içerisinden salt kaynakların eldesi büyük önem teşkil etmektedir. Bu sayede karışım halindeki kaynak verileri tekil olarak elde edilip analiz edilebilir ve karışımı oluşturan unsurlar çok daha net olarak anlaşılabilir. Örneğin, sismik verilerde bunun uygulanması, deprem anında merkez kaynakla eş zamanlı fakat daha az hareketlilik gösteren daha zayıf merkezlerin daha çok sayıda ve daha isabetli olarak tespitlerini sağlayabilir. Kokteyl parti probleminin çözümü, kalabalık ortamlarda şüpheli ve suçlulara karşı, kolluk kuvvetlerince yürütülen teknik takiplerde hedef şahısları dinlemeyi kolaylaştırabilir. Radyo dalgalarında yapılacak bir kaynak ayrıştırması, radarların daha akıllı çalışmasını, daha fazla hedefi daha isabetli izlemesini sağlayabilir.

1985'ten beri bu probleme karşılık yapılan çalışmalarda, zamanla başarımların arttığı, birçok çözüm önerisi ortaya atılmıştır.

Bu alanda farklı amaçlarla fakat aynı problem üzerinde yapılmış birçok çalışma mevcuttur. J.-F. Cardoso, çok boyutlu veri içindeki bağımsız bileşenlerin çıkarılması için basit bir cebirsel yöntem sunmuştur [3]. J.-F. Cardoso ve B. H. Laheld, "Eşdeğişkenli adaptif kaynak ayrıştırma" adlı çalışmalarında Bağımsızlıkla Eşdeğişkenli Adaptif Ayrıştırma (EASI) adlı, eşdeğişkenli tahminin bir adaptif versiyonunu uygulayan kaynak ayrıştırma için bir grup adaptif algoritmalar sunmuşlardır [4]. A. Belouchraniler, kaynak sinyallerinin zaman uyumundan faydalanan yeni bir kaynak ayrıştırma tekniği ortaya koydular [5]. A. Taleb ve C. Jutten, doğrusal olmayan karışımlar içinde karşılıklı bağımsız kaynakların ayrıştırılması üzerine çalışmışlardır [6]. Tzyy-Ping Junglar, Bağımsız Bileşen Analizi (ICA) ile, kör kaynak ayrıştırma tabanlı, elektroensefalografi (EEG) kayıtlarından, geniş çeşitlilikte kusurları ayıklayan, yeni ve genel kullanıma uygun bir yöntem önermişlerdir [7]. Sam T. Roweis, "refiltering" (tekrar filtreleme) adında, tek kayıttan alt bantların frekanslarını dinamik olarak maskeleyerek, kayıt içinden kaynakları elde eden bir teknik sunmuş ve bu maskeleme fonksiyonunu öğrenmek üzere istatistiksel algoritmaların uygulanmasını önermiştir [8]. N. Linh-Trunglar, Zaman-Frekans-Kararsız Kör Kaynak Ayrıştırma (TF-UBSS) algoritmasını alt uzay projeksiyonuyla birleştiren, kesişim noktalarının belirgin olarak işlenmesi imkanını sunan, yeni bir algoritma oluşturmuşlardır [9]. E. Vincentlar, genel Kör Ses Kaynağı Ayrıştırma (BASS) problemlerine uygulandığında, algoritmaları değerlendirmeye ve karşılaştırmaya yardım eden yeni bir nümerik performans kriteri tasarlamışlardır [10]. Huan Taolar, geçici tahmin edilebilirliği maliyet fonksiyonu olarak kullanan bir adaptif kör kaynak ayrıştırma yöntemi önermişlerdir [11]. S. Santosh Kumarlar, Otomatik Konuşma Tanıma (ASR) sistemlerinde istenmeyen sinyallerden kaynaklı performansı düşüren hataları ortadan kaldırmak için, negatif olmayan faktörizasyon metodunu kullanarak, kaynak ayrıştırma yöntemi için yeni bir yaklaşım sundular [12].

Son otuz yıllık süreçte, problem için sunulan ve sürekli geliştirilen yaklaşımlarda, bileşen analizleri ve matematiksel açılımlar ağırlıktaydı ve tüm modeller varsayımlar üzere tasarlanmaktaydı. Bu süre boyunca problemin doğası daha derin irdelenip anlaşıldıkça, probleme dönük olarak belirli senaryolarda daha isabetli çözüm yaklaşımları sunmaya başlandı. Son on yıllık süreçte, bu problem üzerine yapılmış çalışmalarla geliştirilen yüksek başarımlı yaklaşımlar ile birlikte, birçok (yazılımsal) araçlar geliştirilmiştir. Bu zaman diliminde ağırlıklı olarak istatistiksel analizler, kümeleme yöntemleri, işlevsel ve nümerik algoritmalar, örüntü işleme ve yapay sinir ağlarını merkez alan öneriler ortaya çıkmıştır. Son beş yılda ise, sinir ağlarının ağırlığının daha da arttığı, derin öğrenme metodolojisinin de bu

alana katılmasıyla birlikte başarımların yükseldiği ve çözüm sürecindeki insan etkileşiminin azaldığı çalışmaların ağırlıkta olduğu görülmektedir.

Güncel olarak, kaynak ayrıştırma, kokteyl partisi problemi, bozulmuş verinin onarılması, karışmış verilerin ayıklanması ve stil transferi gibi konular yeni çalışmalarda öne çıkmaktadır. Biz de çalışmamızda farklı kaynaklardan oluşan bir karışımın, bu karışımı oluşturan kaynaklara ayrılmasının istendiği bir kokteyl partisi problemi senaryosu üzerinde duracağız.

2 İlgili Kavramlar

2.1 Tarihçe

Derin öğrenmenin geçmişi 1943'e kadar uzanır. Bu yılda Walter Pitts ve Warren McCulloch insan beynindeki sinir ağlarını temel alan bir hesaplama modeli oluşturmuşlardır [13]. "Eşik mantığı" adını verdikleri, matematik ve algoritmaların bileşkesinden oluşan bir sistem ile düşünce sürecini taklit etmeyi hedeflemişlerdir. Bu yıldan itibaren Derin Öğrenme alanı istikrarlı şekilde evrimleşmeye devam ederken, bu süreçte iki önemli duraklama geçirmiştir. Ve bu duraklamaların ikisi de, kötü şöhrete sahip Yapay Zeka Kışları'nda yaşanmıştır.

1940'larda canlılarda düşünce süreci üzerine çalışmak için geliştirilen "bağlantıcılık" düşünce ekolü ortaya çıkmıştı. 1950'de Alan Turing düşünen bir makinenin nasıl test edilebileceğine dair önerisini sunduğu bir makale yazdı [14]. İnancı şuydu ki, eğer bir makine telem tarzında, bir insanı fark edilemeyecek farklarla taklit ederek, bir sohbeti devam ettirebiliyorsa, o makinenin düşündüğü söylenebilir. Bu makaleyi 1952'de beynin Hodgkin-Huxley modeli takip etti [15]. Nöronların elektriksel bir ağ oluşturduğu, her bir nöronun ikili mantıkla (on/off) çalıştığı bir modeldi. Bu olaylar, 1956'da Dartmouth Koleji'nin sponsorluğundaki bir konferansta, yapay zeka kavramının ortaya çıkışının fitilini ateşledi.

1958'da Frank Rosenblatt basit toplama ve çıkarma işlemlerini kullanan, iki katmanlı hesaplayıcı sinir ağını baz alan, örüntü tanıma için bir algoritma olan Perceptron'u tasarlamıştır [16]. Bu kavram ilerleyen yıllarda kullanılarak ve üzerine yeni yaklaşımlar geliştirilerek, sinir ağlarındaki işlevsel derinliğe ciddi katkı sağlanacak ve sinir ağlarının problem çözmedeki etkinliği artırılacaktır. Kendisi matematiksel notasyonların olduğu ek katmanlar önermiş olsa da, 1975'e kadar bu gerçekleştirilememiştir.

1960'da Henry J. Kelley "Optimum Uçuş Yollarındaki Meyil Teorisi" başlıklı makalesiyle, Geri Yayılma Modeli'nin temellerini atmıştır [17]. 1962'de sadece zincir ku-

ralı üzerinden, daha basit bir versiyonu, Stuart Dreyfus tarafından geliştirilmiştir [18]. 1965'te derin öğrenme algoritmaları üzerine ilk çalışmalar, Veri Yönetiminin Grup Metodu'nu geliştiren Alexey Grigoryevich Ivakhnenko ve "Sibernetik ve Tahmin Teknikleri" yazarı, Valentin Grigor'evich Lapa'dan gelmiştir. Polinom (karmaşık denklemler) aktivasyon fonksiyonlu modeller kullanıp, elde edilen sonuçlar istatistiksel analize tabi tutulmuştur. Her katmandan, istatistiksel olarak en güçlü öznetlikleri seçilerek sonraki katmana gönderilmiştir. Her ne kadar sinir ağlarına derinlik katan ve daha sonuç odaklı algoritmalar ortaya konmuş olsa da, süreçteki istatistiksel seçilim görevi insana ait olduğundan, günümüzdeki algoritma ve modellere göre genel olarak çok daha yavaştır.

Yapay zekanın gelişimi verimsizdi ve gelişmeye başladığı zamanla uyumsuzdu. Matematik, algoritma alanlarında çok ciddi ilerlemelere ve o zaman için astronomik güçte bilgisayarlara ihtiyaç vardı. 1956'da heyecan verici, hayal kurduran bir kavram olarak başlayan yapay zeka alanına verilen araştırma destekleri 1970'lerde, ilerlemedeki yetersizliği eleştiren birkaç rapordan sonra kesilmeye başlandı. İnsan beynini taklit etme çabasının o zamanki sonuçları olan sinir ağları üzerinde deneyler yapıldı ve çalışmalar rafa kaldırıldı. En başarılı örnekler sadece basit problemleri çözebiliyordu, bu modellerden etkilenmeyenlerce de oyuncaklar olarak tanımlanıyordu. Yapay zeka araştırmacıları hedeflerine ulaşma konusunda fazla iyimserlerdi, ve karşılaşabilekleri problemler hakkında safça varsayımlarda bulunuyorlardı. Söz verdikleri sonuçlar gerçekleşemeyince, desteklerin kesilmesi ve ilginin kaybolması süpriz değildi. Böylelikle 1974'de ilk yapay zeka kışı başladı. Ve direkt ilgili olduğu sinir ağları ve derin öğrenme araştırmaları da aynı derecede etkilendi, gelişim durdu. Bu duraklama 1980'e, "Uzman Sistemler"in piyasaya çıkışına kadar sürdü.

1980'lerde Birleşmiş Milletler ve İngiltere, Japonya'nın yeni "beşinci jenerasyon" bilgisayar projesi ile rekabet etme ve bilgisayar teknolojisi alanında dünya lideri olma hedefiyle, araştırmaları yeniden desteklemeye başladı. Ayrıca 1970'lerde popüler bir yapay zeka araştırma konusu olan Uzman sistemler'in de, 1980'lerde yazılım endüstrisine girişi ile yapay zeka alanı yeniden hareketlenmeye başladı. Bu sistemler ilgili konudaki uzman bilgileri eşliğinde, kural seti ve eşik değerleri biçiminde, nitel ve nicel sınırlamalarla, göreve özel tasarlanmış, gelişmiş otomasyon programları olarak endüstriye yeni teknoloji kapıları araladılar, fonksiyonelliğe, esnekliğe, iş gücüne, süreç yönetimine, otomasyona ciddi katkı verdiler ve şirketlere büyük mali tasarruf sağladılar. 1987'de gelecek ikinci yapay zeka kışına kadar yapay zeka alanı gelişimine, büyük oranda uzman sistemler üzerinden devam etti. Bu süreçte bağlantıcılık ekolünden de faydalanıldı.

Yapay zeka alanı 1987'den 1993'e kadar bir diğer büyük kışı yaşadı. Yapay zeka araştırmalarındaki bu ikinci yavaşlama, o dönemde piyasanın en güçlü uz-

man sistem çözümü XCON'un zamanı ile çakıştı. Diğer uzman sistem bilgisayarları yavaş ve güvensiz görülmeye başlanmıştı. Çünkü masaüstü bilgisayarlar git gide popülerleşmekte ve daha eski, daha iri ve çok daha az kullanıcı dostu bilgisayar bankalarının yerini almaktaydı. Sonunda, uzman sistemlerin yönetimi, masaüstü bilgisayarlara göre basitçe çok pahalılanmaya başladı. Güncellenmesi zordu ve öğrenemiyordu. Masaüstü bilgisayarlar bu problemleri yaşatmıyordu. Yine aynı dönemlerde Savunma Gelişmiş Araştırma Proje Ajansı ($DARPA$) yapay zekanın dünyada yeni dalgayı oluşturacak güçte olmadığı kanısındaydı ve desteklerini daha çabuk sonuç alacağını düşündüğü projelere yönlendirdi. Sonuç olarak 1980'lerin sonlarında yapay zeka araştırmalarına verilen destek büyük ölçüde kesilmiş ve ikinci yapay zeka kışı başlamıştı.

Kunihiko Fukushima 1980'de hiyerarşik ve çok katmanlı yapay sinir ağı olan Neocognitron'u tasarladı [19]. Bu yapı sibernetiğin yapay zeka alanına verdiği en önemli katkılardan biridir ve sanal örüntü tanıma uygulamalarında kullanılmaktadır. 1989'da derin sinir ağlarını kullanan algoritmalar geliştirildi. Fakat dönemin bilgisayarları ile ağ eğitim süreleri günler ölçeğinden başladığı için gerçek dünya uygulamalarına uygun değildi. Yüksek performanslı grafik işlem birimleri ortaya çıkana dek teorik çalışmalar devam etmesine karşın performans yetersizliği nedeniyle derin sinir ağları üzerine fazla uygulama yapılamadı ve araştırmacıların bu konudaki ilgisi oldukça düşüktü. Fakat yapay sinir ağlarının gelecek vaat etmediği düşünüldüğü bu durgun süreçte yaşanan bilimsel gelişmeler, performans gereksinimi ortadan kalktığında derin öğrenme ve yapay zeka alanlarında yaşanacak patlamaya ve gelecek sıçramalara zemin hazırlamıştır. Yine bu yılda Yann Le Cun geri yayılımın ilk pratik gösterimini sunmuştur. 1992'de Juyang Weng birbirinden ayrı görüntülerinden üç boyutlu nesne tanıma metodu olan Cresceptron'u yayımladı [20].

1999'da ilk Grafik İşlem Birimi (GPU) üretildi ve sundukları paralel işlem kabiliyeti, hesaplama hızlarını onlarca kat artırdı. Daha sonra bu oran 2000'lerin ortalarında yüzlere, 2010 sonrası ise binlere çıkınca ikinci yapay zeka kışını yaşatan performans engelini açık farkla aşmış ve yapay zeka ile birlikte derin öğrenme kavramı da yapay sinir ağlarından ayrı ve özel bir dal olarak ilgi görmeye başlamıştır. Yüksek performanslı bilgisayarlar eşliğinde, çok daha fazla deneyler ve analizler yapılmış ve derin ağ tasarımları hızlı şekilde gelişmiştir. META Grup'un 2001'de veri türü ve kaynaklarının çeşitliliğine, verinin artan hacmine ve verinin hızlı erişilebilirliğine dikkat çekerek Büyük Veri (Big Data) kavramının ortaya çıkışına öncülük etmiştir. Geoffrey Hinton ve Ruslan Salakhutdinov 2000'lerin ortalarında çok katmanlı sanal ağların, nasıl katman katman ön eğitime tabi tutulabildiğini gösterdikleri bir makale yayımladılar. Bu makale derin öğrenme terimi lehine, ilerleyen yıllarda fırtınaya dönüşecek rüzgarı başlattı ve derin öğrenme hem endüst-

ride hem de bilim camiasında hızla popülarite kazanmaya başladı. 2009'da Fei-Fei Li, herkesin erişimine açık bir veritabanı olan ve 14 milyon başlıklı resimlerden oluşan ImageNet'i duyurdu [21]. Li, büyük verinin makine öğrenmesinin çalışma şeklini değiştireceğini öne sürdü. Verinin, öğrenmenin temel faktörü olduğuna dikkat çekti. 2009'da yapılan, konuşma tanıma için derin öğrenme üzerine yapılan NIPS çalıştayında, yeterince büyük bir veri kümesi ile, sinir ağlarının ön eğitime ihtiyacı olmadığı ve hata oranlarının önemli ölçüde düştüğü saptandı [22].

2011'de GPU performansının geldiği noktada, artık konvolüsyonel sinir ağlarının katman katman ön eğitimini yapmadan konvolüsyonel sinir ağlarını eğitmek mümkün hale gelmişti. Ve bunun ilk örneği olan Alex Net ortaya çıktı [23]. 2011 ve 2012'de uluslar arası yarışmalar kazandı. 2012'de Google, Kedi Deneyi olarak bilinen projesinin sonuçlarını açıkladı [24]. Bu serbest araştırma projesi, denetimsiz öğrenmenin zorluklarını keşif için yola çıktı. Sonrasında bu temel soruna karşı sunduğu, denetimli öğrenme üzerine kurulan yaklaşım örneği gösterdi ki; yapay örüntü tanıma algoritmaları, artık belli görevlerde insan düzeyinde performansa ulaşmıştı. 2016'da tamamlanan Google DeepMind'ın algoritması AlphaGo, karmaşık masa oyununda ustalaştı ve o yılın dünya ikincisi olan profesyonel Go oyuncusu Lee Sedol'u, Seoul'da halka açık bir turnuvada yendi. [25] [26] [27]

2.2 Derin Öğrenme

Derin öğrenme, sayısallaştırılabilen geniş kapsamda problemlerin çözümü için ortaya atılmış, algoritma ve topolojilerden oluşan bir metodolojidir. Yeni olmamasına karşın son yıllarda gösterdiği patlamanın sebebi, paralel işlem performansının, hızla gelişen grafik işlem birimleri (GPU) ile artması ve sonucundaki aktifleşme ile birlikte Büyük Verilerin (Big Data) de bu alanı beslemesidir. İçerdiği çözümleme yaklaşımları, temel makine öğrenmesi yaklaşımlarının aksine, tekil sınıflandırma esasına değil, çoklu sınıflandırma esasına dayanır. Bu metodolojide tasarlanan ağın genelde çok katmanlı yapıda olması, hiyerarşik tasarımı ve ağ içi görev paylaşımı itibariyle insan algısını taklit ettiği gözlenir.

2.3 Yapay Sinir Ağı

İnsanlar ve diğer hayvanlar bilgiyi sinir ağları ile işlerler. Bunlar birbirleri ile elektrik sinyalleri vasıtası ile iletişim kuran trilyonlarca nörondan (sinir hücresi) oluşmaktadır. Bu biyolojik yapıları taklit eden algoritmik veya nümerik yapılar, yapay sinir ağı olarak adlandırılır.

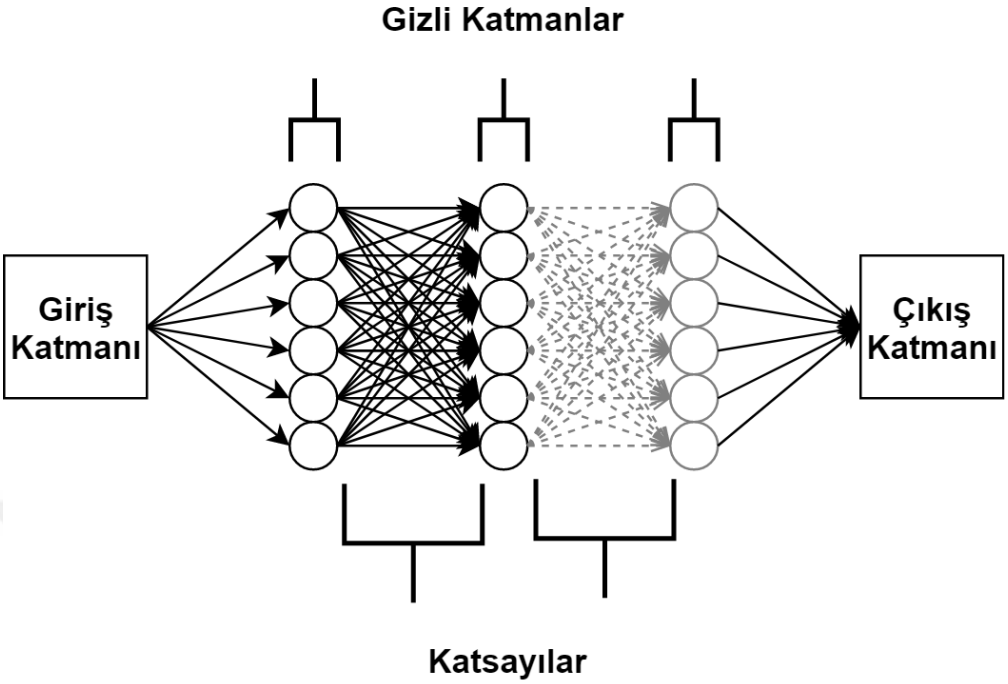
Sinir ağı araştırmaları iki temel amaç üzeredir: insan beyninin daha iyi anlaşılması, ve tıpkı insanlarda olduğu gibi, soyut ve iyi tanımlanmamış problemleri çözebilecek bilgisayarların geliştirilmesi. Bu doğrultuda sinir bilimcilerin insanın sinir sistemi ile ilgili elde ettiği yapısal bilgilerin temel alındığı modellerinden, matematikçilerin insan algısına ve zihnine hipotetik yaklaşarak tasarlanan matematiksel modellerine kadar, birçok sinir ağı yapısı denenmiştir [28]. Yapay sinir ağları genel olarak, nümerik veri ve fonksiyon içerebilen hücrelerin bir arada bulundukları, katman adı verilen yapıların birbirine bağlanmasıyla oluşur.

2.3.1 İleri Beslemeli Sinir Ağları

İleri beslemeli sinir ağı (FNN), yapay sinir ağlarının en basit şekillerinden biridir. Şekil 2.1'de örneği verilmiştir. Veri, giriş katmanından girip çıkış katmanından çıkar. Giriş katmanından, çıkış katmanına, tek yönlü bir hareket vardır. Sadece giriş ve çıkış katmanlarından da oluşabilir, bu katmanlar arasında gizli katman denilen, ağ boyunca akan verinin üzerinden doğrudan veya işlenerek geçtiği katmanlar da bulunabilir.

2.3.2 Tam Bağlı Sinir Ağları

Tam bağlı sinir ağı ($FCNN$), katmanlar arası geçişlerde, katmanlardaki her bir hücrenin, komşu katmandaki her bir hücreye bağlı olduğu bir sinir ağı yapısıdır. Şekil 2.1'de bir örneği görülmektedir. Bu yapının avantaji, verinin katmanlar arası geçişleri esnasında, her bir katmandaki geçişin tam bağlı olması suretiyle, katmanlar arası her bir hücrenin, komşu hücrelerle ilişkisinin temsil edilebilmesidir. Ayrıca yeterli veri örneği ile eğitilip, kullanılan verinin doğası üzerinden önemli ve isabetli değişken ilişkilendirmeleri saptanabilir.



Şekil 2.1: İleri beslemeli ve tam bağlı sinir ağı ($FFCNN$) genel yapısı

2.4 Yapay Sinir Ağı Mimarisi

Yapay sinir ağı mimarisi, tasarlanacak bir yapay sinir ağında kullanılacak hücreler, katmanlar, fonksiyonlar ve algoritmalar ile birlikte tüm yapay sinir ağının, yapısal, işlevsel ve süreçsel tasarımının bütünüdür. Günümüzde farklı amaçlar için özelleştirilmiş farklı mimariler mevcuttur. Derin öğrenme metodolojisinin ve alanının genişlemekte olduğu zamanımızda, sürekli olarak, var olan mimariler de kapsadığı alt birimleri gibi, daha büyük ve daha kapsamlı yapay sinir ağı yapılarına modüler olarak entegre edilebilmektedir. Getirilen yeni yaklaşımlar ve çözümler potansiyel yeni mimarilerin ortaya çıkmasına yol açmaktadır.

2.4.1 Otokodlayıcı (Autoencoder)

Otokodlayıcı ideal senaryoda giriş ve çıkışın aynı olduğu özel bir yapay sinir ağı mimarisidir. Otokodlayıcı giriş verisinin çok alt düzeylerdeki temsilini, bir başka deyişle daha yüksek derecede esas teşkil edebilecek unsurlarını, güçlü bir özet halinde öğrenebilir. Sonra bu alt düzeydeki temsili, yeniden şekillendirilerek, orjinal giriş verisi çıktı olarak üretilir. Temel görevi girişin tahmin edildiği bir istatistiksel ilişkilendirme probleminin çözümlenmesidir. Bu ağın merkez katmanı, ağdaki en dar katmandır ve gerekli eğitim sonucunda, verinin en alt düzeydeki ve en belirleyici özelliklerinden oluşan bir özetini içerir. Giriş ve çıkış katmanları en geniş katmanlardır ve merkeze doğru daralır. Şekil 2.2'de mimarinin genel yapısı gösterilmiştir.

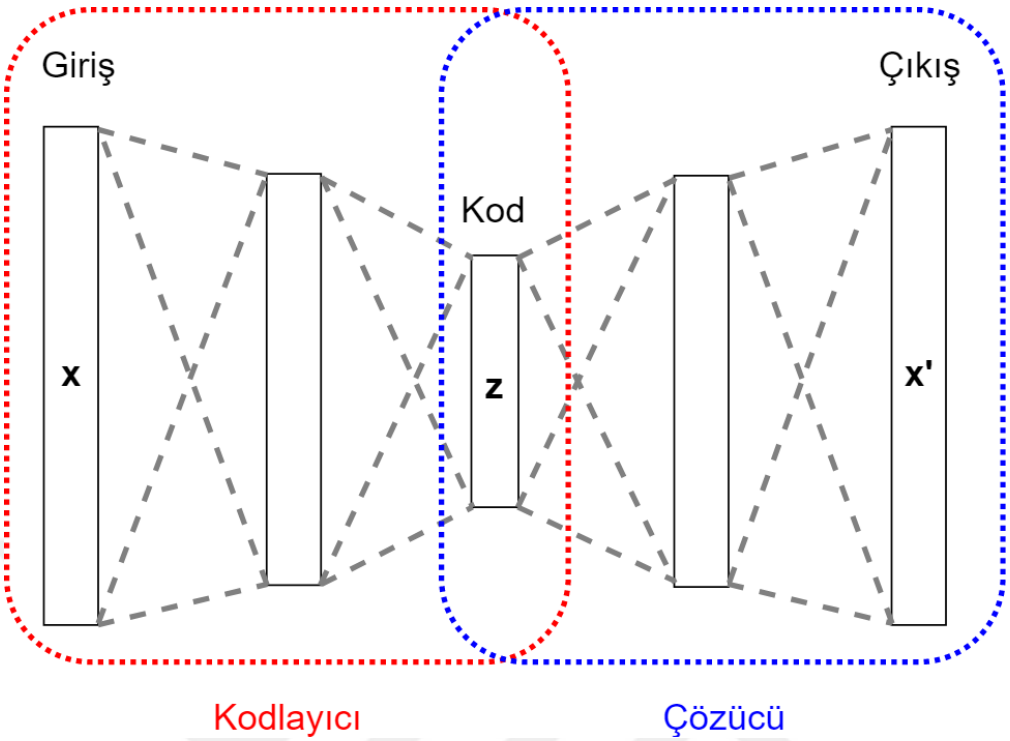
Tipik bir otokodlayıcı mimarisinin üç ana bileşeni vardır:

Kodlayıcı: Kodlayıcı, her aşamada hücre sayısı düşen katmanlardan oluşur. Giriş katmanından, merkez katmana kadar devam eder. Giriş verisinin isabetli ve kompakt bir özetini çıkarma görevini üstlenir.

Kod: Merkez katmanın kendisidir. Verinin alt düzey ve sıkıştırılmış halini içerir. Veri küçülür fakat kalitesi artar.

Çözücü: Kodlayıcının tam zıttı bir yapıdadır. Her aşamada hücre sayısı artan katmanlardan oluşur. Merkez katmanından, çıkış katmanına kadar devam eder.

Kodlayıcının ürettiği koddan (merkez katman), orjinal giriş verisini tahmin ederek (görüntüsünü) yeniden sentezleme görevini üstlenir. Bu vasfıyla aynı zamanda kodlayıcının aynası konumundadır. Başarılı bir çözme için, başarılı bir koda ve dolayısıyla başarılı bir kodlamaya ihtiyaç vardır. [29]



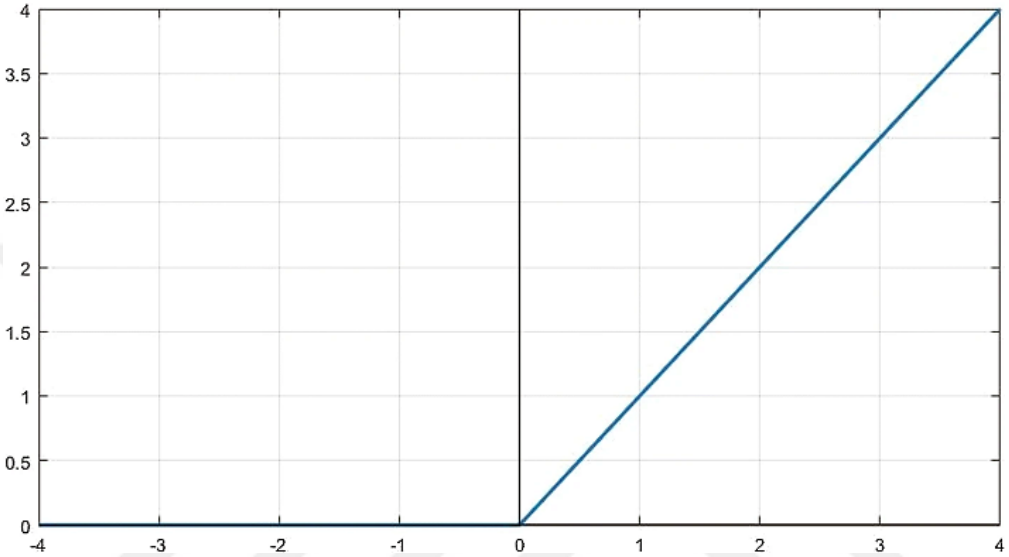
Şekil 2.2: Otokodlayıcı mimarisinin genel görünümü

2.5 Doğrusal Birimler

Doğrusal birimler, yapay sinir ağının eğitim sürecinde karşılaştığı kaybolan düşüm problemine (vanishing gradient problem) çözüm olarak ve yapay sinir ağında doğrusal olmayan aktivasyon fonksiyonlarının hesap yükünü azaltmak için tasarlanmış, taklit ettiği doğrusal olmayan fonksiyona en yakın çoklu değer karşılıklarını içeren ve çok sayıda doğrusal fonksiyondan oluşan ve gereği halinde doğrusal olmayan fonksiyonlar da barındırabilen bileşke bir yapıdır. Bu birimler kullanılarak, ağın öğrenmesi daha kolay kontrol edilebilir, daha fazla öğrenmesi sağlanabilir ve görece düşük bir hesap hata payını indirgemek için ortaya çıkan büyük işlem yükü minimuma indirgenebilir. En yaygın kullanılan örneği doğrultulmuş doğrusal birimdir ($RELU$).

$$h(x) = \lambda \left\{ \begin{array}{ll} x & if\ x > 0 \\ 0 & if\ x \leq 0 \end{array} \right\} \quad (1)$$

Şekil 2.3'te aktivasyon fonksiyonu verilmiştir. Bu örnekte giriş parametresi sıfır ve daha küçük iken çıkış sıfır, giriş parametresi sıfırdan büyük iken çıkış, parametrenin kendisidir.

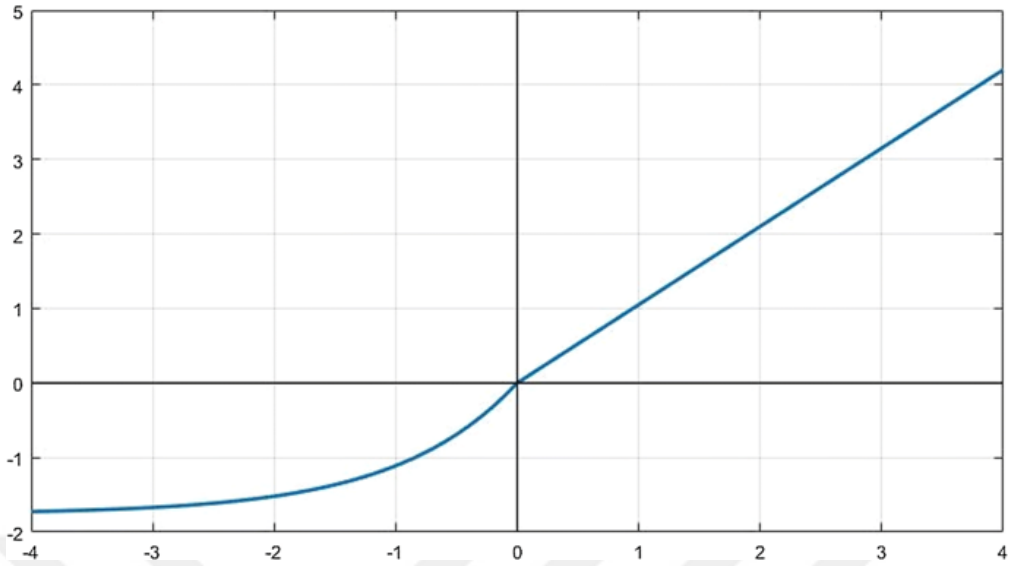


Şekil 2.3: *RELU* aktivasyon fonksiyonu [30]

2.5.1 SELU aktivasyon fonksiyonu

$$h(x) = \lambda \left\{ \begin{array}{ll} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \leq 0 \end{array} \right\} \quad (2)$$

Ölçeklendirilmiş üstel doğrusal birim ($SELU$), $RELU$'nun aksine; sıfırdan küçük değerlerde de çıkış değeri verir. Sıfırdan küçük değerler negatif ve doğrusal olmayan özel bir fonksiyondan geçerler. Sıfırdan büyük olan girişler RELU'da olduğu gibi yine değişmeden çıkarlar. Önerildiği makalede [31] yapılmış parametre ayarlamalarıyla, içinden geçirilen veri kümesinin sıfırda merkezlenmesini ve normalizasyonunu (sıfır etrafında dağılan ve sapması bir olan veri kümesi) sağlayan bir doğaya kavuşmuştur. Bu halinin örneği Şekil 2.4'te gösterilmiştir.



Şekil 2.4: $SELU$ aktivasyon fonksiyonu ($\alpha \simeq 1.6733$ and $\lambda \simeq 1.0507$) [30]

2.6 Hata Metrikleri

Hata metrikleri, ilgilenilen verinin gerçek zemin kabul edilen bir referans kritere göre hata oranını ifade etmek için oluşturulmuş ölçülerdir. Optimizasyonun, büyük verinin, veri kümelerinin, istatistiğin ve sinir ağlarının olduğu her yerde bu metriklere ihtiyaç duyulmaktadır.

2.6.1 Ortalama Kare Hata

Ortalama kare hata (MSE), gerçek zemin ve tahmin edilen olmak üzere, birbirine denk iki veri kümesinin belirli olduğu, tahmin edilen verinin gerçek zemin, yani orjinal veriden sapmasını veya referans veriye kıyasla hata oranını hesaplar [32]. Dolayısıyla birimi de yoktur. Bu hesabı bu iki veri kümesi arasında birbiriyle öznetlik anlamında eş nitelikte olan iki farklı nicelik arasındaki farkların karelerinin aritmetik ortalamasını alarak yapar.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i\prime) \quad (3)$$

Formülünün doğası gereği değeri daima pozitiftir. En iyi senaryo değerinin sıfır olmasıdır, ki bu da hata veya sapma olmadığı anlamına gelmektedir. MSE, hatanın

ikici derecesi olduğundan, tahmin değeri ile ilgili, hem sapmayı hem de kaymayı içerir.

2.6.2 Zirve Sinyal-Gürültü Oranı

Zirve sinyal-gürültü oranı ($PSNR$), bir sinyalin olası en yüksek değerinin, kalitesini bozan distorsiyon değerine oranını ifade eder [33]. Distorsiyon ise MSE ile hesaplanır. Desibel (dB) cinsinden ifade edilir.

$$\begin{aligned} PSNR &= 10 \cdot log_{10} \frac{MAX_i^2}{MSE} \\ &= 20 \cdot log_{10} \frac{MAX_i}{\sqrt{MSE}} \\ &= 20 \cdot log_{10} - 10 \cdot log_{10} MSE \end{aligned} \tag{2.1}$$

Bu metrik, ilgili verinin, orjinal referansına oranla distorsiyonu üzerinden bu distorsiyon altında verinin orjinaline göre ne kadar koruyabildiğini veya bu veriyi ne düzeyde içerdiğini ortaya koyar.

3 Çözüm Önerisi

Çözüm önerimiz bir kaynak ayrıştırma problemi de olan, kokteyl partisi problemi üzerinden oluşturulmuştur. Gerçek hayatta karşılaştığımız çoğu işitsel ortam, aynı anda ses üreten çok sayıda unsur içerir. Kokteyl partisi problemi hedeflediği çözüm ile bu unsurların, işitsel ortamda duyulan tüm seslere ait karışımdan ayrılmasını amaçlayan bir kaynak ayrıştırma yaklaşımıdır. Kokteyl partisi probleminde çözülmesi gereken iki ana problem vardır. Bunlardan ilki, sesi ayrıştırmaktır. Karışımdan ayrılan her bir sesin, birbirinden tamamen ayrı olarak yeniden oluşturulabilmesi gerekmektedir. Ana problemlerden ikincisi ise, ilgilenilen kaynak dışındaki kaynak verisi anlık olarak ihmal edilirken, ilgilenilen kaynağın devamlı takibidir. Ses ayrıştırma probleminin başarılı bir şekilde çözülebilmesi, bu problemin çözülmesine bağlıdır.

Problemin çözümlenmesi, öncelikle karışım üzerinde frekans ayrıştırması yapılmasını gerektirir. Sonrasında problemin çözümlenmesi ile ilgili iki temel problem ortaya çıkar. Birincisi, karışım içindeki ses parçalarının hangi kaynağa ait olduğunun bilinememesidir. İkincisi ise karışımın belli yerlerinde, kaynakların birbirinin enerjisini daha yoğun enerjiyle bastırarak maskelemesidir. İkisini de doğuran temel neden karışım ve kendisini oluşturan kaynaklar arasında ortak olarak fiziksel tutarlılık bulunmasıdır. Burada farklı kaynakların çözümlenebilmesi için karışım üzerinde bölümlendirme yapılmalıdır. Bu işlemde karışım üzerinde ilgilenilen kaynağa ait sesin mümkün olduğunca dinleyici tarafından belirgin işitilebilen ve maskelenmemiş kısımları toplanıp ilgilenilen kaynağın karakteristiği saptanmaya çalışılır ve elde edilen veri maskelenmiş kısımların ayrılabilmesi yahut tahmin edilebilmesi için kullanılır.

Probleme dönük sunduğumuz çözüm için derin sinir ağlarından ve derin öğrenme metodolojisinden faydalanmaktayız. Kaynak ayrıştırmaya dönük örnek senaryomuz için, farklı kaynaklar ve bu kaynaklarla oluşturulacak karışım verilerine ihtiyaç vardır. Bu veriler ayrıştırıcı sinir ağımızın eğitilmesinde ve başarımının ölçülmesinde kullanılır. Ayrıştırma işlemi eğitimde öğretilmiş ve sürecin başında belirlenmiş kaynaklar ve bu kaynaklardan oluşan karışımlar üzerinden, denetimli olarak

yapılır. Bu doğrultuda kaynak olarak ses kaynakları üzerinde durmaktayız. Bulunan sesler çoğunlukla doğal işitsel ortamlarda kaydedilmiş olduğundan, seçilen verilerde tekil bir kaynağın ses verisi içinde azami derecede baskın olması tercih sebebidir.

Derin sinir ağımızın kaynak ayrıştırmaya uygun bir mimari ile oluşturulması gerekir. Başarımı artıracak en güçlü etkenler, veri üzerinde çalışırken ilgilenilen kaynaklara odaklanabilme yeteneği ve veriye ait yüksek öneme sahip kısımları saptayarak bu verileri öne çıkarma yeteneğidir. İlk etken farklı kaynaklara ait verilerin karışım içerisinden çıkarılması için, ikinci etken ise farklı kaynaklara ait karakteristiklerin saptanarak maskelenme etkisinin asgari düzeye indirgenmesi, bu sayede karışım içinde fiziksel tutarlılığın zorlaştırdığı yerlerde ayrıştırmanın desteklenmesi için önemlidir.

Sunduğumuz çözüm için öncelikli olarak ne tip ses verisi alınacağının belirlenmesi ve seçilen verilerin programlama ortamına eklenmesi gereklidir. Daha sonra eklenen verinin işleme hazır hale getirilmesi için ön-işlemlere tabi tutulmalı ve eğitiminde kullanılacağı sinir ağına uyumlu olacak şekilde yeniden düzenlenmelidir. Bu işlemler yapılırken ihtiyaç duyulan verinin, özü itibariyle korunması önem arz eder.

Tasarladığımız ağı, topladığımız veri ile eğitirken, elimizdeki senaryoya uygun eğitim parametreleri belirlememiz gerekir. Eğitim sürecindeki optimum kayıp değerlerine ulaşmak için optimizatör kullanılır. Eğitim sürecinin başarımı doğrulamalar ile kontrol altında tutulur. Eğitim verisi dışında belirlenen bir test verisi ile eğitilen ağ sistemine yabancı bir karışım dinletilir ve yapacağı ayrıştırma üzerinden başarımı değerlendirilir. İdeal senaryomuz, ayrıştırıcının öğrendiği kaynakların, karışım içinden olabildiğince birbirinden bağımsız birer ses kaynağı olarak çıkmasıdır.

Tüm bu çözüm tasarısının uygulanacağı bir programlama ortamına ihtiyaç duyulmaktadır. Bu ortamda kullanılacak, ses verisi okuyup yazabilecek, yapay sinir ağları oluşturup eğitebileceğimiz, ilgili tüm görevleri yürütürken gerekli temel matematiksel işlev ve yapıları sağlayacak kütüphanelere de ihtiyaç duymaktayız.

3.1 Verinin Temini

Öncelikle sistemimizde ihtiyaç duyulan ses verisi ihtiyacını karşılamak için mümkün olduğunca tekil ses kaynağı verisi bulacağımız internet arşiv ve paylaşım platformlarının araştırılması gerekir. Aranan veriler bulunduktan sonra bu verilerin çözümümüzde denetimli eğitim verisi olarak kullanılmak üzere bir multimedya

düzenleme yazılımı ile düzenlenmesine geçilir. Karışım da bu aşamada üretilir çünkü problemin çözülmesine dönük eğitim ve bu eğitimin başarımının ölçülebilmesi için bir zemin gerçeğe ihtiyaç vardır. Bu yüzden karışımı elde olan tekil kaynaklardan düzenleyerek oluşturmamız, problemin çözümünü önceden bilmemiz sayesinde, bize bu zemin gerçeği sağlar. Karışımdan önce düzenlenen tekil kaynaklar ise, örnek üzerinde belirlenen kriterlere göre düzenlenip, sistemimizde kullanılmak üzere son halini alır. Sonrasında düzenlediğimiz bu tekil kaynaklara ait karışım düzenlenir ve sistemimiz için veri girişine hazır hale gelir. Bu noktada ses verilerini oluştururken verinin çıktısı en saf haliyle, sıkıştırılmamış ses dosyası formunda verilir.

3.2 Ön İşlemler

Hazırladığımız ses dosyaları içinden istediğimiz ses verilerini almamız, daha sonra işlenecek ses kanalına ait değerleri alıp işlenmek üzere sistemimize dahil etmemiz gerekmektedir. Bunun için öncelikle çalışma ortamımızda ses dosyalarını okuyup yazabilecek bir kütüphaneye ihtiyaç vardır. Bu kütüphaneye ait sınıf ve fonksiyonlar kullanılarak önce dosyalar okunur ve içeriği değişkenlere aktarılır. Daha sonra dosya ile ilgili alınan bilgiler içerisinden ses verisi ve ses verisinin içinden ilgilenilen ses kanalına ait veri dizisi alınıp yeni bir değişkene aktarılır.

Oluşturduğumuz veri dizilerine ait değişkenlerin sayısı her örnek için, iki tekil kaynak ve bu iki kaynağın oluşturacağı karışım olmak üzere, üç tanedir. Daha sonra bu üç dizi için en uygun işleme parametreleri belirlenerek ön işleme sokulur ve eğitim sürecine hazır hale getirilir. İşleme parametrelerimiz, dizi içinden kullanılacak veri uzunlukları, veriyi belirlenen uzunlukta eşit parçalara bölecek olan vektör boyu ve bu iki değişkene bağlı olarak belirlenen vektör sayısıdır. n veri uzunluğu ile seçilen $nx1$ ölçülerindeki veri dizimiz, her bir elemanı, belirlenen $mx1$ ölçüsüde vektörler olan bir vektör dizisi halinde; $(n/m)xm$ boyutlarında bir iki boyutlu dizi olarak yeniden düzenlenir ve vektör sayısı sayısal değerini taşıyan vektör kümemiz eğitim için işleme hazır hale gelir.

Veri uzunlukları, örnek sayısı üzerinden, alınan veri uzunluklarını ifade eder. Bu parametreler ile aldığımız verinin ne kadarının eğitim ve doğrulama (eğitim fonksiyonunda, veri kümesi üzerindeki, doğrulamaya ayırma parametresi ile birlikte) işlemlerinde, ne kadarının ise test işleminde kullanılacağını belirleriz. Eğitime ayrılan miktarın artması öğrenmeyi güçlendirir, fakat doğrulamaya ayrılan miktarı azaltarak bu yapılırsa, eğitim esnasında ağın hatasını değerlendirmek güçleşir. Teste ayrılan miktar ise ağın başarımını sağlıklı olarak değerlendirebilecek kadar uzun olmalıdır, eğer eğitime ayrılan miktar azaltılarak bu yapılırsa, ağın eğitim

başarımı düşer. Vektör boyu bizim veriyi işlemedeki etkileşim penceremizdir. Nihai olarak veri uzunluğu ile birlikte bir vektör kümesi oluşturur ve bu kümenin vektör sayısını belirler. Vektör kümesinin sayısal değeri veri kümesinin zenginliğini, vektör boyu ise vektör kümesindeki her bir vektörün içerdiği detayı (çözünürlük) örnek sayısı cinsinden ifade eder. Kısa vektör daha çok sayıda ve daha az detaylı örneklerden oluşan bir vektör kümesi oluşturur. Uzun vektör ise daha az sayıda fakat daha fazla detay içeren vektör kümesi oluşturur. Buradaki temel fark, kısa ve uzun vektörlerin farklı örnekleme senaryolarıyla uyumlarıdır. Çok sayıda ve net sayısal değerlere anlık ihtiyaç duyulan bir ağda vektör yerine direkt olarak noktalar (1 uzunluklu vektör) bile kullanılabilirken, anlamlı örüntüleri öğrenmeye ihtiyacı olan sistemlerde uzun vektörler kullanılabilir. Ağa uygun olan vektör boyu kısa ise uzun vektör boyu öğrenme sürecinde ezberlemeye yol açabilir. Eğer ağa uygun olan vektör boyu uzun ise kısa vektör boyu öğrenme sürecinde kararsızlığa yol açabilir. Her iki parametre de bu kriterler göz önüne alınarak en uygun şekilde seçilmelidir.

3.3 Ayrıştırıcı Ağ Yapısının Tasarımı

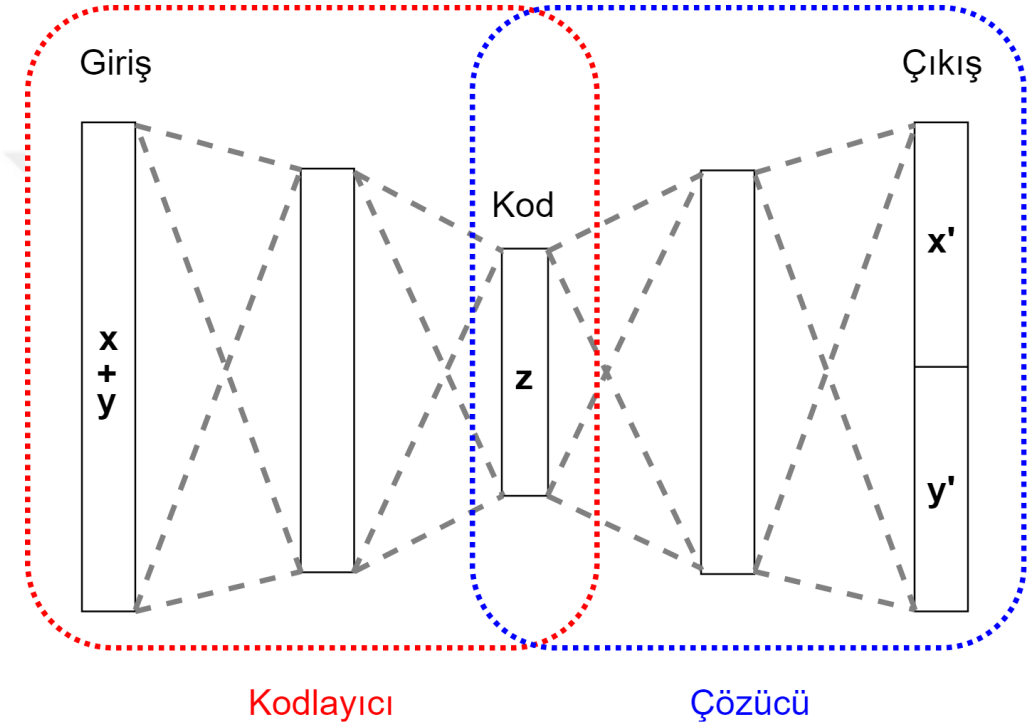
Ses karışımını ayrıştırmak için eğitilecek ve sonrasında ayrıştırıcı işlevde kullanılması hedeflenen ağ yapısı tasarımı, çözümün çekirdeği konumudadır. Kullanılacak sinir ağındaki katmanların ve tüm ağın yapısı, öncelikli olarak verinin uyumlu şekilde ağa girişini, sonrasında eğitim ve test başarımını, son olarak daha sonra yeniden sese dönüştürülebilecek çıktıların veri yapısı korunarak alınmasını, tümüne paralel olarak performansı ve verimi öncelemelidir.

Ağa ses karışımı olarak verilen tekil veri girişine karşılık, karışımı oluşturan sesleri ayrı ayrı temsil eden, birden fazla sayıda veri çıkışına ihtiyaç duyulmaktadır. En iyi performans için ayarlanabilecek parametreler, sayısal olarak, katmanlara ait veri giriş ve çıkış boyutları, kalıp alt ağ yapılarına ait özel işleme parametreleri, yapısal olarak ise ağ tipleri ve dizilimleridir.

3.3.1 Otokodlayıcı Ayırıcı Modelin Yapısı

Ağ mimarimiz otokodlayıcı mimarisine benzerdir. Fakat temel farkı, girişten tekrar girişin görüntüsünü üretmek yerine, iki orjinal verinin karışımından, orjinal verilerin görüntülerini üretme görevini yüklenmiştir. Bu bağlamda çıkış boyu da giriş boyunun iki katıdır. Bu farklılık Şekil 3.1'de gösterilmiştir. Ağımızda boyutsal indirgemeden, özetleme kabiliyeti itibarı ile faydalanmaktayız. Bu kabiliyet bize hem güçlü öznetliklere odaklanma imkanı sağlar hem de özet fazlası öznetliklerin atılmasıyla çalışma süresini düşürerek performansta iyileşme sağlar.

Model ağımız ileri beslemeli tam bağlı katmanlardan oluşur. Katman çıkışlarında işlenmek istenen verinin doğasına göre doğrusal, ReLU veya SeLU aktivasyon fonksiyonları kullanılabilir. Katman topolojisi ve parametreleri otokodlayıcı mimarisinden esinlenilen ayırıcı modelimizin, giriş katmanından orta ara katmana kadar, katmandan katmana, katman boyu küçülür. En küçük boydaki orta ara katmandan çıkış katmanına kadar ise katmandan katmana, katman boyu yeniden uzatılarak, çıkış katmanında karşılık geleceği orjinal vektör boyutlarına çıkarılır. Burada otokodlayıcı mimarisinin özetleyici niteliğinden faydalanarak daha net bir ayrıştırmayı hedeflemekteyiz.



Şekil 3.1: Ayrıştırıcı sinir ağı mimarisinin genel görünümü

3.4 Eğitim Yaklaşımı

Eğitime hazırlanmış olan veri giriş ve çıkış katmanlarımız üzerinden sinir ağımıza bağlandıktan sonra sıradaki işlem, girişteki karışım verisini, çıkışa orjinal tekil kaynak verilerini yerleştirmek suretiyle, tekil kaynaklara ayırmak üzere eğitmektir.

Sinir ağı yapısı ve çalıştırılacak parametrelerine bağlı olarak tasarlanacak eğitim sürecinin, kendisine giriş katmanında verilen ses karışımını, çıkış katmanında tamamen sesi oluşturan orjinal kaynaklara ayırmayı hedeflemesi gerekmektedir. Bu

sürecin işleyişinde sinir ağının katman düzeni,katmanlar arası uyum, veri düzeninin sinir ağı katmanlarıyla ve veri ilişkilendirmeleriyle uyumu, süreçsel işleyişin veri ve ilişkilendirmeler üzerindeki etkisi, eğitimde kullanılacak verilerin eğitim sürecindeki yeterlilik derecesi, eğitim başarımında belirleyici unsurlar olmaktadır. Performans optimizasyonu için bu unsurlar, ağırlık güncelleme aralığı (örnek vektör grup büyüklüğü) ve tüm eğitim kümesinin ağa eğitim için beslenmesi suretiyle tekrar sayısıdır.

Eğitim sürecindeki temel parametrelerimiz, giriş, çıkış, doğrulama ve test verilerimiz ve bu verilerin çeşitli dinamikleri, çevrim sayımız ve grup büyüklüğümüzdür. Grup büyüklüğü parametresi, ağımızdaki çarpanların (ağırlıkların) kaç örnekte bir güncelleneceğini belirlemektedir. Bu örnekler de verinin önceden yeniden düzenlenmesi ile hazırlanmış vektörlerdir. Grup büyüklüğünün küçük veya büyük olmasının eğitim sürecine etkisi, vektör boyununun kısa veya uzun olmasının yaptığı etki ile aynı olmasa da benzerdir. Grup büyüklüğü farklı olarak verinin eğitimde kullanılması noktasında vektör boyuna göre bir üst düzeydedir. Vektör boyu ile veri eğitim için uygun parçalara bölündükten sonra, grup büyüklüğü ile kaç vektörün bir araya gelerek anlamlı bir değişime yol açabileceği belirlenir. Vektör boyu ağın tanımlama yönelimini kontrol etmek için, grup büyüklüğü ise sınıflandırma yönelimini kontrol etmek için önemlidir. Çevrim sayısı ile eğitim sürecinin kaç kez tekrarlanacağı belirlenir. Bu sayede hem eldeki eğitim verisinden daha iyi istifade edilebilir, hem de eğitim tekrarlarla kuvvetlendirilebilir. Gereğinden az olması kararsızlığa, gereğinden fazla olması ise ezberlemeye yol açabilir. Senaryoya göre optimum değerler değişebilir ve belirlenen kriterler çerçevesinde ideal değerler bulunup parametreler ayarlanmalıdır.

3.4.1 Otokodlayıcı Ayırıcı Modelin Eğitimi

Katman topolojisi ve parametreleri otokodlayıcı mimarisinden esinlenilen ayırıcı modelimiz, giriş katmanından orta ara katmana kadar, katmandan katmana, katman boyu küçültülerek, önceki katmandan gelen çarpan vektörünün daha net ayırt edici öznetliklerin ağırlıkta olduğu bir özeti alınır. Orta ara katmandan çıkış katmanına kadar ise katmandan katmana, katman boyu yeniden uzatılarak, mevcut özetten üretilecek çıkışta karşılık geleceği orjinal vektör boyutlarına çıkarılır. Bu sayede eğitim için alınan veri üzerinde belirleyici öznetliklere odaklanılır ve ilgili aktivasyonlara etkisi daha zayıf olan bilgilere karşılık gelen çarpanların etkisi daha da zayıflatılarak, sinir ağının belirlenen öznetlik tabanlarında ayırt ediciliği güçlendirilmiş olur. Özetleyicilik niteliği, aynı zamanda kısıtlı veya kısmen bozuk veriler ile daha iyi sonuçlar da sağlayabilir.

3.4.2 Optimizasyon

Eğitim sürecimizde çarpanlarımızın optimum değerlere ulaşması amaçlanmış olsa da basedilen metodolojinin tek başına uygulanması halinde işlem yükü görece alternatif metotlara oranla çok daha yüksek olmaktadır. Bu noktada optimizasyon metotları devreye girer. Algoritmik metotlar üzerinde çalışan optimizatörler, işlem yükünü büyük ölçüde azaltarak, *PC*'lerde tolere edilebilecek düzeylere indirir. Eğitilen ağda yakınsanılan değerlere optimizatör yardımıyla ulaşılarak, en ideal değerler ve en az kayba (loss) ulaşılmış olur. Kayba ulaşma oranına müdahale edilerek öğrenme oranının kontrolü de sağlanabilir.

3.4.3 Doğrulama

Eğitimde optimizatör hedef değerlerine ulaştıktan sonra prosedür öncesinde eğitim verisi içinden ayrılarak eğitim amaçlı kullanıma kapatılmış bir veri kümesi ile doğruluk oranları belirli bir hata ölçüsü ile saptanır. Böylelikle veri seti üzerindeki eğitimin, aynı veri üzerindeki başarımı görece ortaya çıkmış olur. Test verisi ile doğrulama verisini ayıran en önemli fark, doğrulama verisinin eğitim süresince defalarca kez kullanılabilmesine karşın, test verisinin yalnızca modelin eğitimi bittikten sonra kullanılabilmesidir. Test verisi tamamen eğitilen ağa yabancı olmalı, ağ kendi öğrendikleriyle test verisini çözümlemelidir. Buna karşın doğrulama verisi, eğitim esnasında başarımımızı kontrol altında tutmak amacıyla kullanılır. Ve kullanıldıkça ağ bu verileri daha çok ezberleyebilir. Bu da zamanla doğrulamanın güvenilirliğini azaltır. Bu yüzden doğrulama ve teste ayrı ayrı ihtiyaç vardır.

3.5 Test

Eğitilmiş sinir ağımız artık ayırıcı görevine hazırdır. Tahmin işlemi hem eğitilmiş ağın işlevi için kullanılmak üzere yeni veri girişleri yapılarak bu girişlerin çıktılarının alınmasına, hem de bu suretle eğitilmiş ağın ayırt ediciliğinin test edilerek gerçel başarımının ortaya konmasını sağlar. Bunu sayısal ifadeye dönüştürmek için belirli bir benzerlik ölçüsü kullanılabilir. İdeal senaryoda sonuç olarak, girilen karışımdan, karışımı oluşturan kaynaklar, tekil düzeyde ayrılmış olarak elde edilir. Testte amacımız eğitmek değil, eğitimini tamamlamış bir modeli sınamaktır.

4 Metodoloji

4.1 Çalışma Ortamının Oluşturulması

Sistemimizi oluşturduğumuz, ses verisinin toplandığı, işlendiği, sinir ağımızı oluşturduğumuz ve verimiz ile çalıştırdığımız, çalışma ortamı rolünü de üstlenen programımız, bu konularda basit, etkili ve yaygın olarak kullanılan Python [34] programlama dilinde yazılmıştır. Programlama için tercih ettiğimiz çalışma ortamı, (Google) Colaboratory olmuştur [35]. Google'ın sunduğu bu bulut hesaplama hizmeti, herkese açık ve ücretsizdir. Colaboratory kodlama için ön uç olarak açık bir proje olan Jupyter Notebook [36] tabanlı bir düzenleyici ve çekirdeği olan IPython'u [37] kullanmaktadır [38]. Programlama da bu düzenleyici ile yapılmaktadır. Donanım kaynağı olarak ise işlemci ve bellek ile birlikte tensör işlemede GPUlara oranla çok daha hızlı çalışan TPUlar sunulmaktadır. Sinir ağlarının oluşturulması, yönetilmesi ve işletilmesi için Tensorflow [39] kütüphanesi üzerinde geliştirilmiş olan Keras [40] kütüphanesi kullanılmıştır. Nümerik işlevleri ve farklı boyutlarda dizi değişkenleri için numpy kullanılmıştır [41]. Verileri okuyup yazmak için depolama amaçlı olarak da, pydrive [42] ve Colaboratory ortamına ait entegre google.colab kütüphaneleri kullanılarak, Colaboratory üzerinden yine açık ve ücretsiz olan (Google) Drive'ın [43] sunduğu bulut depolama alanı programımıza bağlanmıştır.

4.2 Verinin Temini

Sistemimizde ses örnekleri olarak kullanılmak üzere *wav* (Waveform Audio) dosyaları tercih edilmiştir [44–47]. Ses materyallerimiz, YouTube [48] video paylaşım platformu üzerinden indirilmiş videoların ve Cornell Üniversitesi'nin kuş türlerine ait bilgileri depoladığı ve paylaştığı, ornitoloji veri tabanındaki [49, 50] ses örneklerinin önce seçilmesi, sonra video düzenleyici yazılımı DaVinci Resolve [51] ile videoya ait sesin içinden parçaların seçilip derlenmesi ve son halinin *wav*

dosyası olarak çıktısının alınması süreciyle elde edilmiştir. Wav dosyalarımızın her biri Doğrusal Vuruş Kod Modülasyonu (LPCM) [52, 53] ile çift kanal, 16-bit çözünürlükte ve 48kHz frekansta örneklenerek oluşturulmuştur. Sisteme girişi yapılmaya hazır bu dosyalar, Soundfile [54] kütüphanesi kullanılarak, işlenmek üzere çalışma ortamımıza eklenmiştir.

4.3 Ön İşlemler

Ortama eklenmiş olan ses dosyalarımızın her biri ortama, veriyi içeren bir dizi ve ses dosyasına ait taban frekans değerinden oluşan bir ikili olarak giriş yapmaktadır. Elde ettiğimiz bu veri içinden istediğimiz ses verisini almamız, daha sonra işlenecek ses kanalını alıp işlenmek üzere sistemimize dahil etmemiz gerekmektedir. Öncelikli olarak bu iklinin içinden dizi çekilir. Çok boyutlu dizi içinden işlenecek kanala ait dizi seçilerek alınır. Sonrasında elde ettiğimiz, şiddet değerlerinden oluşan veri dizimize ait işleme parametreleri belirlenir. Bu parametreler, dizi içinden kullanılacak veri uzunlukları, veriyi eşit uzunlukta örneklere ayıracak vektör boyu ve bunlara bağlı olarak belirlenen vektör sayısıdır. Bu parametreler eşliğinde diziden seçilip çekilen veri yeniden boyutlandırılarak $mn \cdot 1$ boyutlu dizi, m boyunda ve n sayıda vektörlerden oluşan $n \cdot m$ matrisi şeklinde yeniden düzenlenir. Böylelikle işlenmeye hazır vektör kümemizi elde etmiş oluruz. Böylece, giriş ve çıkış verilerinin, kendilerine giriş ve çıkış katmanları ile bağlı sinir ağının eğitim süreci için, ön hazırlıkları tamamlanmış olur.

4.3.1 Sinir Ağı Yapısı

Sinir ağımız seri yerleşmiş ileri beslemeli tam bağlı katmanlardan oluşmaktadır. Bu katmanlar keras kütüphanesindeki Dense yapısı kullanılarak oluşturulmuştur. Giriş katmanı giriş vektörünün boyunda, çıkış katmanı ise giriş ile eş boyda iki çıkış vektörünün toplamı boyundadır. Veri kümelerimiz olan, katman boylarıyla uyumlu uzunlukta vektörlere ayırarak yeniden boyutlandırdığımız matrislerimizden her seferinde bir vektör (sütun) giriş katmanında, karşılık geldiği vektörler çıkış katmanında olmak üzere, bu vektörler arasındaki ağırlıklar ve katman çıkışlarındaki doğrusal aktivasyon fonksiyonlarından oluşan ağımız, eğitim sürecince giriş vektörünü, çıkış vektörlerine haritalayacak şekilde güncellenir. Giriş veri matrisindeki her vektör iterasyonu için bu işlem tekrarlanır ve bu şekilde vektör giriş örneği, ve çıkış örnekleri kullanılarak, sinir ağımız eğitilir.

Sinir ağı modelimizi oluştururken, Keras kitaplığındaki Model sınıfından faydalılmıştır. Kullandığımız tanımlama yönteminde giriş katmanından çıkış katmanına her bir katman, kendisine bağlı katman referansları ile birlikte birer değişken olarak tanımlanmıştır. Sonrasında tüm bu değişken zincirinin başı ve sonu olan giriş ve çıkış katmanlarımız, Model fonksiyonuna parametre olarak geçilerek model örneğimiz oluşturulmuştur.

4.4 Eğitim

Eğitimde parametrelerle ağa ait katmanlarda her bir çarpan güncelleme işleminin kaç kez işleneceğini ve tüm bu işlemin kaç kez tekrarlanacağını belirledik. Sonrasında Keras kütüphanesinin Model sınıfı üzerinden oluşturduğumuz örneğimizin erişimine açık fit fonksiyonundan faydalanarak modelimizi eğitim sürecine soktuk. Fit fonksiyonuna parametre olarak, giriş ve çıkış verilerimizi, çevrim sayımızı, grup büyüklüğümüzü ve doğrulama ayrım yüzdemizi girdik. Fit fonksiyonumuz ağımızı girişten çıkışı üretmek üzere eğitirken, grup büyüklüğü kadar örnekte bir güncellemesini yaptı, her çevrim sonunda doğrulamasını yaptı ve bunu çevrim sayısı kadar tekrarladı. Her çevrim sonunda eğitim seti ve doğrulama seti üzerinden hesaplanan hata oranları MSE cinsinden verildi. Modelimizde Adam optimizatörü kullanılmaktadır. Bu optimizatör, optimizasyonun yanı sıra ilgili bir parametre olarak kullandığı öğrenme oranı parametresi ile öğrenme eğrisinin de kontrolünü sağlayabilir.

4.5 Test

Test için model örneğimiz üzerinde yine $Keras.Model$ altındaki $predict$ fonksiyonunu kullandık. Bu fonksiyon fit fonksiyonunda olduğu gibi, parametre olarak örnek veri alır. Ama "fit"in aksine amaç eğitmek değil, sınamaktır. Dolayısıyla "predict" fonksiyonu yalnızca giriş verisini alır ve çıkış verisini sistemin, öğrendikleriyle, kendisi üretmesini bekler. Başarımı ölçmek için, süreç sonucunda fonksiyonun ürettiği orjinal veriye dönük tahmin sonucunun orjinal kaynak verisiyle aralarında PSNR'ı alınır ve test tamamlanır. PSNR bize istatistiksel olarak orjinal verinin ne oranda korunduğunu gösterir.

4.6 Art İşlemler ve Çıktının Alınması

Eğitilen sinir ağının ürettiği tekil kaynak verileri, yeniden ses dosyaları olarak üretilmek için art işlemlerden geçer. Veri matrisi yeniden boyutlandırılarak *wav* dosyasının orjinalinde olduğu gibi diziye dönüştürülür. Frekans değeri ile birlikte yine soundfile kullanılarak, ağ tarafından üretilmiş ve geri dönüşümü tamamlanmış tekil kaynak verileri, *wav* ses dosyası olarak yeniden sentezlenir.

4.7 Deney Sonuçları

Örnek senaryo olarak oluşturduğumuz ağımızda iki hayvan sesinin karışımı, karışımı meydana getiren iki hayvanın sesine ayırma hedefini üstlenmiştir. Kullandığımız Kızılımsı Çalıkuşu [55], Kaplan [56] ve Kangal Köpeği [57] hayvan ses verilerinin her biri düzenlendikten sonra, çift kanal (stereo), 48kHz frekansında, 16-bit çözünürlüğünde, toplamda 17 saniye uzunluğundadır. Bunun 12,75 saniyesi eğitime, 2,25 saniyesi doğrulamaya ve 2 saniyesi ise teste ayrılmıştır. Ağlarımızı denemek için çalıkuşu - kaplan ve kangal - kaplan olmak üzere iki farklı karışım üzerinde durduk. Modelimizi karşılaştırmak için katman topolojisi, katman sayıları ve katman boyları farklılaştırılmış iki model daha oluşturduk ve tüm modelleri isimlendirdik. Modellere ait isimler ve yapısal parametreler Tablo 4.1'de verilmiştir. Deney sonuçlarımızın ses dosyası çıktılarına "https://drive.google.com/drive/folders/1mae7JltEf7_L4hT4aKFbFw4JKO0izu9J?usp=sharing" adresinden erişebilirsiniz.

Ağ İsmi	Katman Dizilimi ve Uzunlukları (Girişten Çıkışa)	Toplam ve Eğitilebilir Çarpan Sayısı
Papyon	500-250-100-50-100-250-500-1000	812250
Balon	500-1000-2500-5000-2500-1000-1000	31513000
Kova	500-600-700-800-900-1000	2904000

Tablo 4.1: Denenen sinir ağlarımızın isimleri, katman topolojileri ve çarpan sayıları gösterilmiştir.

Modelimiz Papyon'un kıyaslandığı Balon modeli, yaklaşımımızın tam aksine merkez katmana doğru genişleyip, uç katmanlarda daralan bir ağdır. Papyon katmanları merkezde giriş katmanının 10'da 1'i boyuna inmekte, Balon ise merkezde giriş katmanının 10 katı boyuna çıkmaktadır. Kova ise giriş katmanından çıkış katmanına doğru katman boyları sabit, giriş katmanının 5'te 1'i, artış gösteren bir modeldir. Üç model de tam bağlı yoğun katmanlardan oluşmaktadır.

Giriş vektör boyu 500, çıkış vektör boyu ise 1000 olarak belirlenmiştir. 1000 uzunluğundaki vektörümüz 500'erli, her biri ayrıştırılan kaynaklardan birini içeren iki vektörü içermektedir. Tablo 4.1'de de görüldüğü üzere dizayn tercihleri, çarpan sayısında da ciddi farklılıklar meydana getirmiştir.

Bu da eğitim süresince işlem birimlerine verilen toplam iş yükünü, veri ile ilgili ağda tutulan detay oranını ve ağın sunulan veriye oranla doygunluğunu ciddi oranda etkileyecektir. Tam bağlı katmanlar arası doğrusal aktivasyon fonksiyonu kullandık.

Eğitim parametrelerimizde doğrulama ayrımını ve grup büyüklüğünü sabit olarak ayarladık. Doğrulama ayrımımız %15 olarak , grup büyüklüğümüz ise 30 olarak belirlenmiştir. Yani eğitim kümesinin tam veya yaklaşık %15'i doğrulama, kalan %85 civarı da eğitim için kullanılır. 30 vektör örneğinde bir de ağ çarpanlarımız güncellenir. Papyon, Balon ve Kova modelleri sırasıyla 8, 7 ve 6 katmandan oluşmaktadır.

Ağ	Çevrim Sayısı	Kayıp (RMS)	Doğrulama Kaybı
Papyon	5	0.0020	0.0009
Balon		0.0020	0.0011
Kova		0.0016	0.0009
Papyon	10	0.0016	0.0008
Balon		0.0042	0.0019
Kova		0.0034	0.0010
Papyon	15	0.0016	0.0008
Balon		-	-
Kova		0.0028	0.0010
Papyon	30	0.0014	0.0008
Balon		-	-
Kova		0.0011	0.0006
Papyon	60	0.0012	0.0007
Balon		-	-
Kova		0.0011	0.0006
Papyon	120	0.0010	0.0007
Balon		-	-
Kova		0.0007	0.0005

Tablo 4.2: Çalıkuşu-kaplan karışımı üzerindeki eğitim sonuçlarımız, farklı çevrim sayıları kadar eğitimleri sonucunda, eğitim seti üzerinden kayıpları ve doğrulama seti üzerinden doğrulama kayıpları, RMS cinsinden verilmiştir.

Çalıkuşu-kaplan karışımımızdaki eğitim sonuçlarına bakıldığında, Papyon'un

eğitim periyotlarının ilk yarısında (5-10-15 çevrim) daha iyi eğitim sonuçları verdiğini görüyoruz. Kova ise bunun akisne ikinci yarıda (30-60-120) daha iyi sonuçlar veriyor. Papyon sınırlı veri kaynağı ile daha iyi iş yaparken, çevrimler arttıkça Kova'nın başarımının istikrarlı şekilde artarak daha başarılı hale geldiğini görüyoruz. Balon ise özellikle 10 çevrim sonucunda da görülebileceği üzere açk farkla geride kalıyor ve sonrasında kayıp 1 değerini aşıyor, yani aynı sürede benzer farklı örneklerde 10-15 çevrim sonrasında çevrimlere devam edildiği takdirde eğitim başarısız oluyor. Bu da aslında Balon modeli için 500 vektör boyunda kullanıldığında verinin yeterince büyük olmadığını ve fazla miktarda çarpanın sınırlı veri üzerinde eğitim başarımı anlamında görece yüksek kayıplarda doygunluğa ulaştığını gösteriyor.

Ağ	Çevrim Sayısı	PSNR	
		Çalıkuşu	Kaplan
Papyon	5	25.823	28.103
Balon		26.215	25.769
Kova		26.043	25.906
Papyon	10	25.869	28.879
Balon		25.408	21.855
Kova		25.703	24.869
Papyon	15	25.875	28.884
Balon		-	-
Kova		26.262	25.113
Papyon	30	26.004	29.441
Balon		-	-
Kova		27.230	28.234
Papyon	60	26.049	30.500
Balon		-	-
Kova		27.249	29.567
Papyon	120	26.129	31.410
Balon		-	-
Kova		27.452	31.181

Tablo 4.3: Çalıkuşu-kaplan karışımı üzerindeki test sonuçlarımız, farklı çevrim sayıları kadar eğitimleri sonucunda, test sonucunun orjinal veriye göre PSNR'ı cinsinden verilmiştir.

Çalıkuşu-kaplan karışımının test sonuçlarına baktığımızda Kova'nın çalıkuşunu, Papyon'un ise kaplanı daha isabetli şekilde yeniden oluşturduğu görülebiliyor. Bunun en önemli sebebi ise bu karışımın düşük frekans bandını ayırt edebilmek için gereken unsurların, veri kümesi içinde daha yüksek yoğunlukta olması ve bunun

sonucu olarak Papyon'un bu banda daha fazla odaklanarak, ortalamada bu frekans bandına daha yakın bir frekans bandı aralığında işlev gösteriyor olmasıdır. Kova ise veri kümesi üzerine olan yorumunu artan katman boylarıyla genişleterek işlemeye devam ettiğinden bariz şekilde daha kolay ayırt edilebilen çalıkuşuna ait yüksek frekans bandını ayırt etmede daha başarılıdır. Fakat buna karşın, daha geniş bir bant aralığında ayırma yapmaya çalıştığından, karışıklığın yoğunlaştığı düşük frekans aralığında ayırma yapmakta Papyon'a oranla daha başarısız olur.

Ağ	Çevrim Sayısı	Kayıp (RMS)	Doğrulama Kaybı
Papyon	5	0.0068	0.0044
Balon		0.0083	0.0050
Kova		0.0073	0.0046
Papyon	10	0.0064	0.0042
Balon		0.0111	0.0064
Kova		0.0073	0.0046
Papyon	15	0.0060	0.0041
Balon		0.0123	0.0067
Kova		0.0070	0.0045
Papyon	30	0.0058	0.0040
Balon		-	-
Kova		0.0066	0.0044
Papyon	60	0.0055	0.0040
Balon		-	-
Kova		0.0058	0.0041
Papyon	120	0.0054	0.0040
Balon		-	-
Kova		0.0053	0.0041

Tablo 4.4: Kangal-kaplan karışımı üzerindeki eğitim sonuçlarımız, farklı çevrim sayıları kadar eğitimleri sonucunda, eğitim seti üzerinden kayıpları ve doğrulama seti üzerinden doğrulama kayıpları, RMS cinsinden verilmiştir.

Kangal-kaplan karışımındaki eğitim sonuçlarında tüm ağlarda kayıpların önemli ölçüde, 2 ila 10 kat arttığı görülüyor. Aynı zamanda ilk çevrimlerden itibaren Papyon'un Kova ile arasındaki çevrimle azalan farkın da, genel başarımın düşmesinin etkisiyle (%25 civarı) küçüldüğü gözleniyor. İki kaynağa ait frekans bantlarının yakınlığının ve bunun yol açtığı maskeleme etkisinin payları bu konuda büyüktür. Öte yandan Papyon'un tüm çevrimlerde, çevrim sayısı arttıkça azalan bir farkla, Kova'yı geçtiğini görüyoruz. Mimarimizdeki ayırt edici öznetliklere odaklanma vasfı, frekans bantlarının birbirine yaklaştığı ve yer yer örtüştüğü bu senaryoda

kendini gösteriyor. Balon ise bahsettiğimiz temel sorunlarından ötürü yine açık farkla sonda kalıyor.

Ağ	Çevrim Sayısı	PSNR	
		Kangal	Kaplan
Papyon	5	21.674	22.262
Balon		21.691	21.147
Kova		21.730	21.841
Papyon	10	22.096	22.306
Balon		20.597	20.469
Kova		21.725	21.717
Papyon	15	22.211	22.333
Balon		20.616	19.863
Kova		21.861	22.022
Papyon	30	22.210	22.440
Balon		-	-
Kova		21.893	22.082
Papyon	60	22.423	22.423
Balon		-	-
Kova		22.210	22.330
Papyon	120	22.337	22.542
Balon		-	-
Kova		22.242	22.356

Tablo 4.5: Kangal-kaplan karışımı üzerindeki test sonuçlarımız, farklı çevrim sayıları kadar eğitimleri sonucunda, test sonucunun orjinal veriye göre PSNR'ı cinsinden verilmiştir.

Kangal-kaplan karışımının test sonuçlarına baktığımızda çalıkuşu-kaplan karışımındaki iddiamızı desteklediğini görüyoruz. Çünkü bu karışımda iki kaynak da yakın frekans bantlarında bulunmakta. Tüm ağların başarımı PSNR üzerinden bakıldığında genel olarak önemli bir düşüş göstermiş. Öte yandan eğitim sonuçlarına paralel şekilde, Papyon'un yine çevrim artışıyla ters orantılı bir farkla Kova'dan önde olduğu görülüyor. Bu noktada yine ilginç bir fark ile karşılaşıyoruz. Kova Kangal üzerinde daha başarılı başlarken, Papyon çalıkuşu-kaplan karışımında olduğu gibi, yine kangal-kaplan karışımında da kaplan üzerinde daha iyi başlıyor, fakat daha sonra her iki kaynak için de Kova'yı geride bırakıyor. Papyon'un kaplana odaklanması ile ilgili iddiamız bir kez daha destekleniyor. Özellikle modelin aktif olduğu frekans aralıklarındaki kaplan sesinin kangal sesi ile yakın frekans aralıklarında olması, ayrıştırılması noktasında da kaplan sesinin kangal sesine göre daha fazla sayıda ve daha keskin değerli öznetliklere sahip

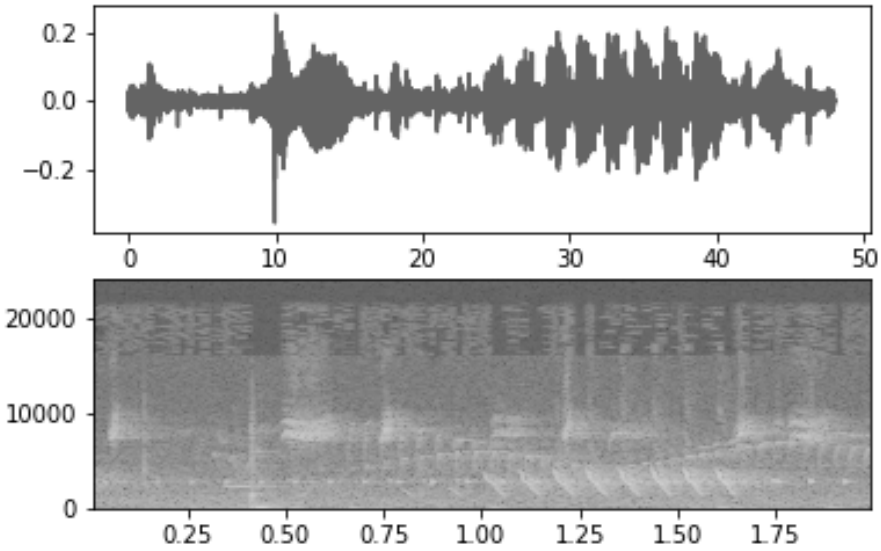
olması ile birlikte kangala karşı sağlayabileceği zıtlık bunda önemli rol oynuyor olabilir.

Ortalama Çalışma Süresi	Ağ		
	Papyon	Balon	Kova
Adım(ms)	0.64775	20	2
Çevrim(sn)			
1 çevrim	1	24	2
5 çevrim	5	120	10
10 çevrim	10	240	20
15 çevrim	15	360	30
30 çevrim	30	600	60
60 çevrim	60	1200	120
120 çevrim	120	2400	240

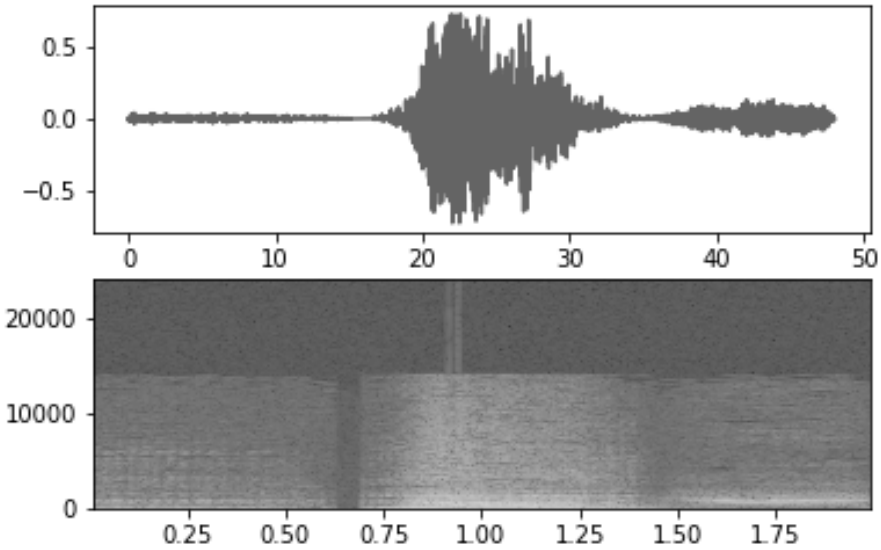
Tablo 4.6: Modellerimizin çalıkuşu-kaplan ve kangal-kaplan karışımları üzerindeki eğitimlerinin toplamı üzerinden ortalama çalışma süreleri yaklaşık olarak adım (bir vektör ile eğitim) süresi ve çevrim süresi cinslerinden verilmiştir.

Eğitim sürelerine baktığımızda Papyon'un bariz üstünlüğü göze çarpıyor. Burada çarpan sayısının ana etkenlerden biri olduğu görülüyor. Diğer ekstrem olan Balon ise ağ ölçeğini büyütmenin maliyetinin bir örneği olarak bu bağlamı destekler nitelikte görünüyor. Kova'nın çalışma süresi ise Papyon'un iki katı civarında.

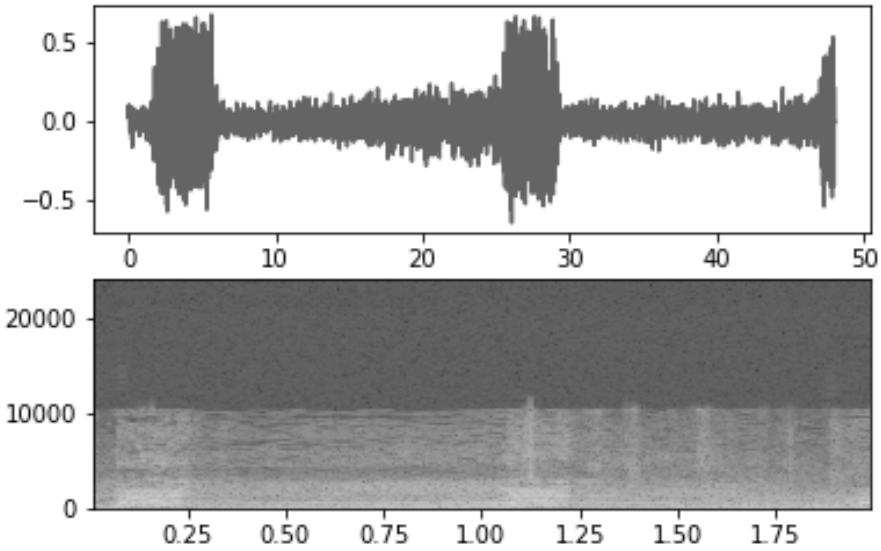
Şekil 4.1-4.17'de kaynaklara ve karışımlara ait test setlerimiz ve eğitilmiş ağlarımızın test çıktılarımızın, genlik-zaman ve spektrogram grafikleri verilmiştir. Balon modelinin eğitim sürecinde erken tıkanması nedeniyle diğer modellerimize nazaran, ayrıştırmaya daha az odaklanabildiği görülüyor. Modellerin eğitim süreçlerince genel olarak veri kümemiz üzerinde frekans yoğunluğunun olduğu zaman pencerelerine ve tekrarın yoğun olduğu frekans aralıklarına odaklanarak ayrıştırmaya çalıştığı her bir kaynağın karakteristiğine yaklaşmaya çalıştığını görüyoruz.



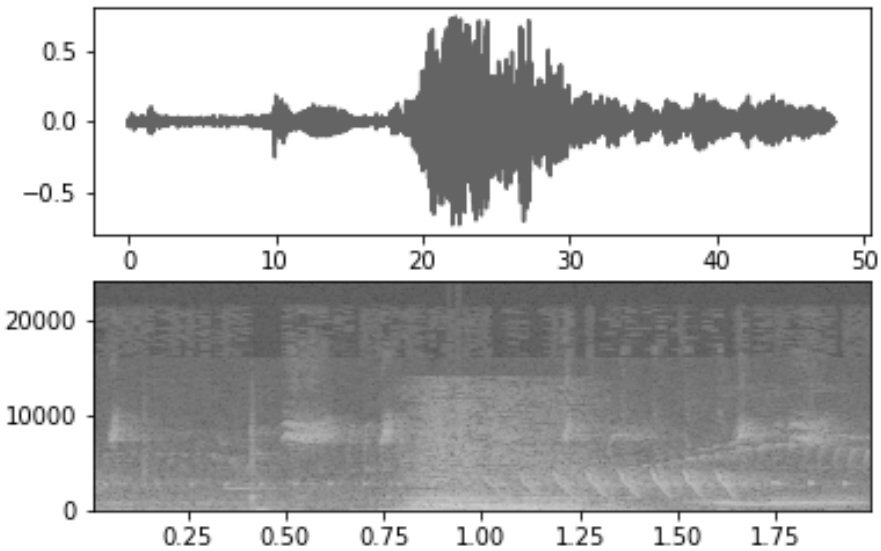
Şekil 4.1: Çalıkuşu test setimizin genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



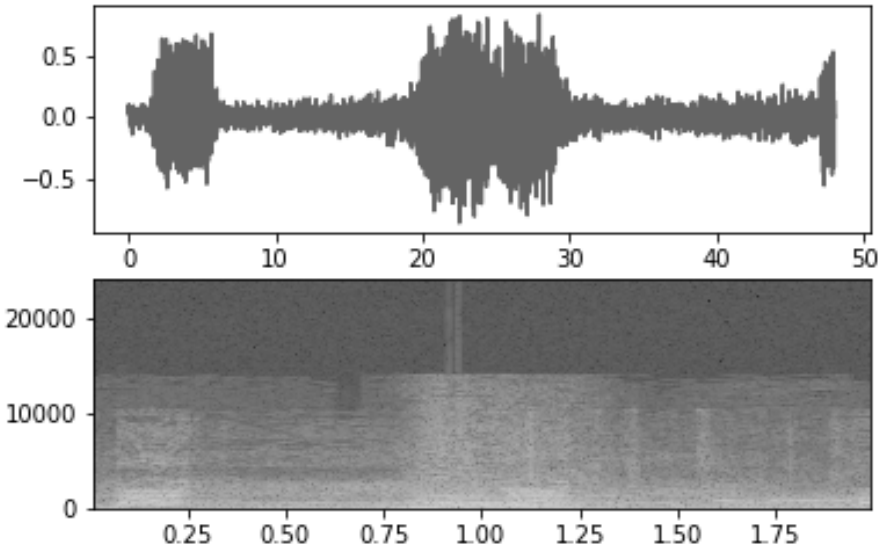
Şekil 4.2: Kaplan test setimizin genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



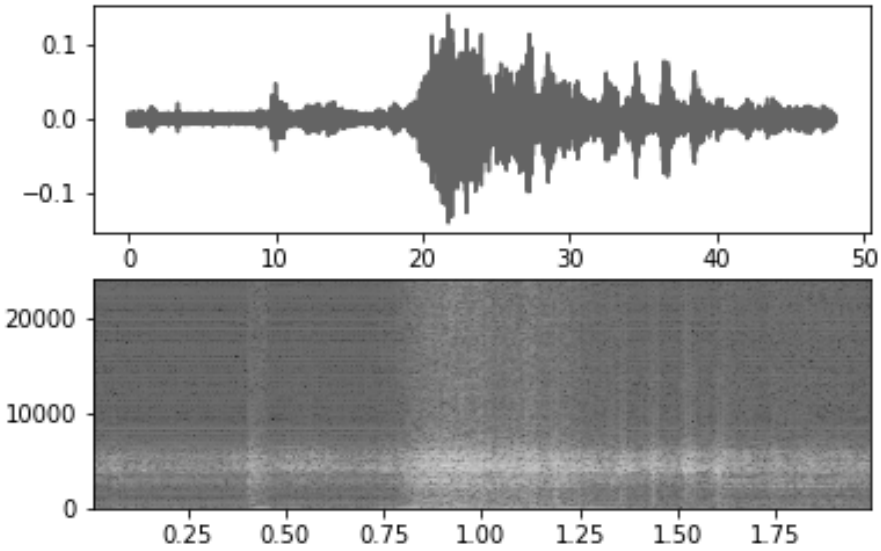
Şekil 4.3: Kangal test setimizin genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



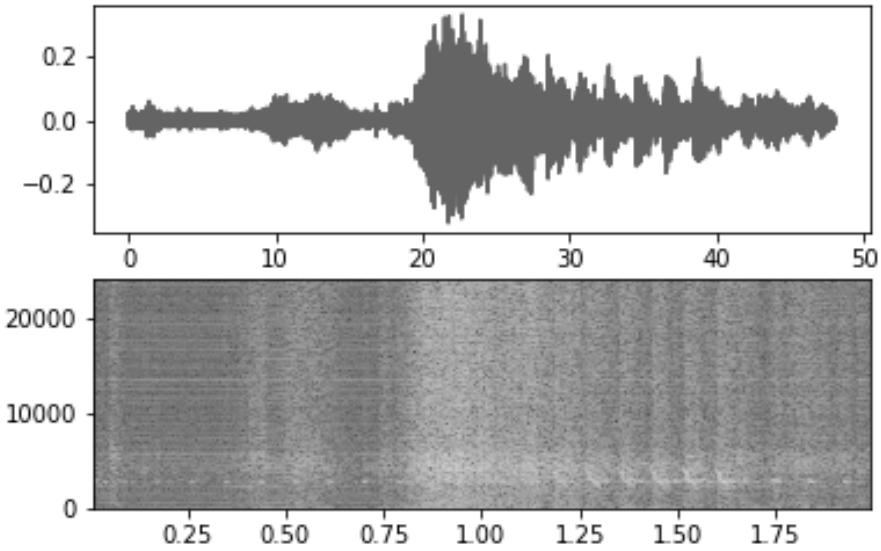
Şekil 4.4: Çalıkuşu-kangal karışımı test setimizin genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



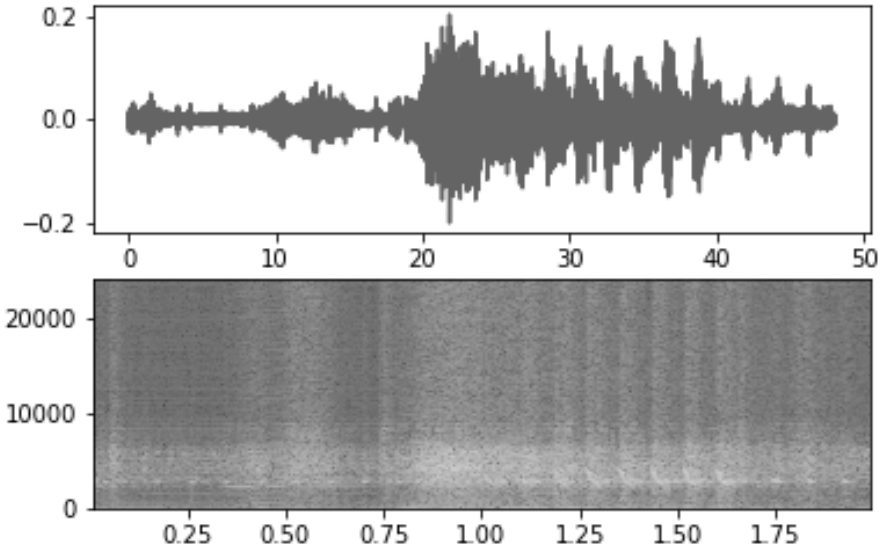
Şekil 4.5: Kangal-kaplan karışımı test setimizin genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



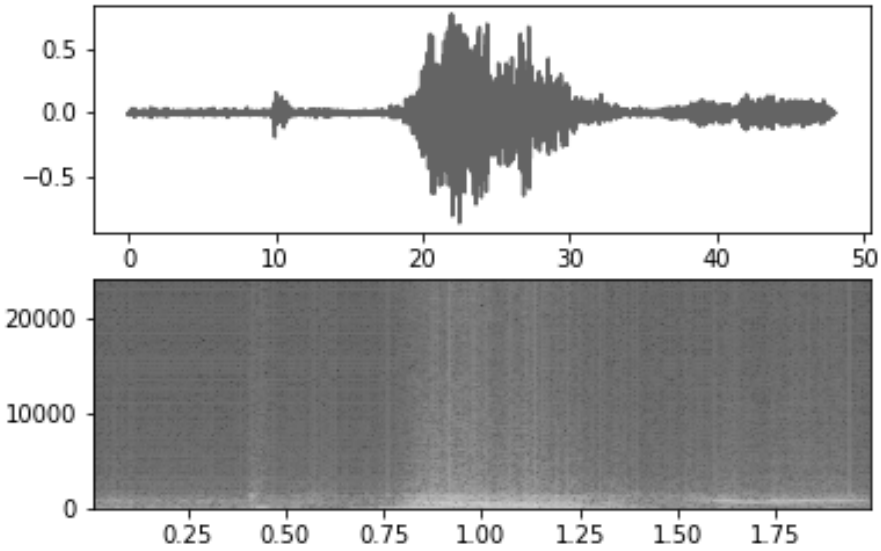
Şekil 4.6: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



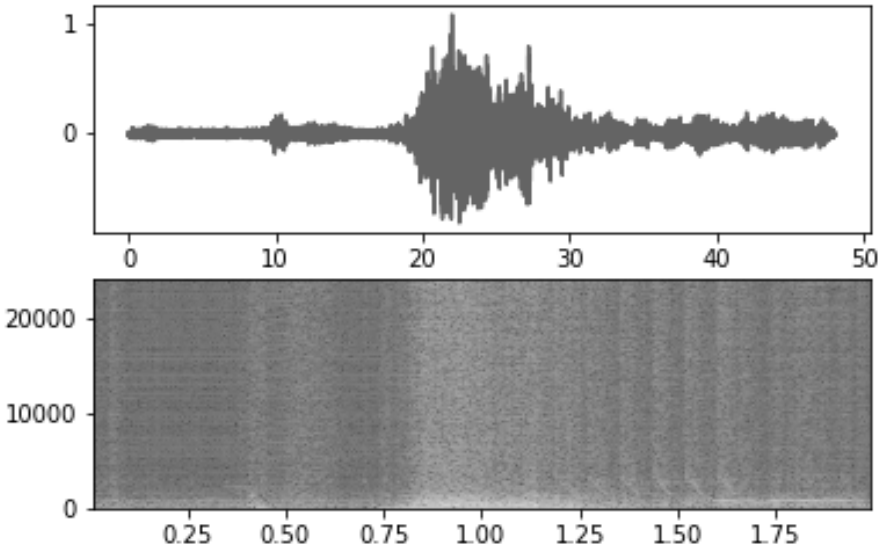
Şekil 4.7: Balon modelinde kaplandan ayrılmış çalıkuşu sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



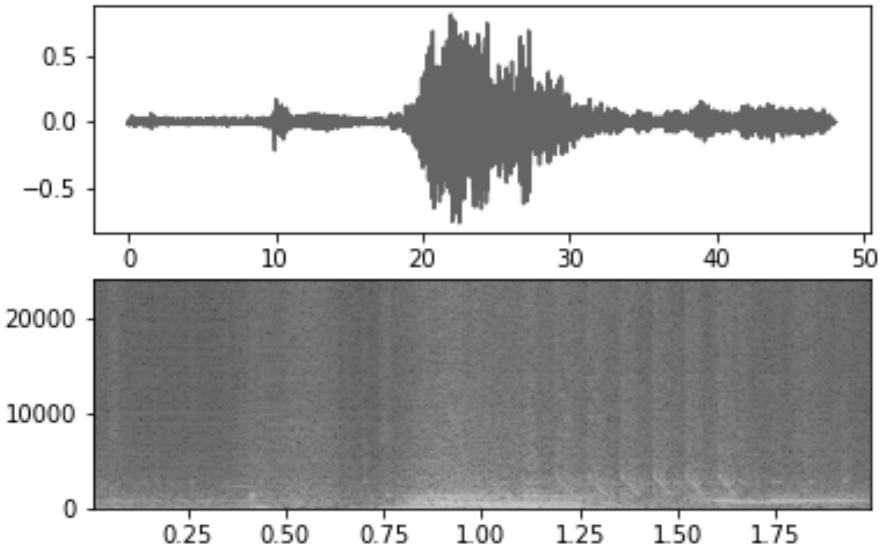
Şekil 4.8: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



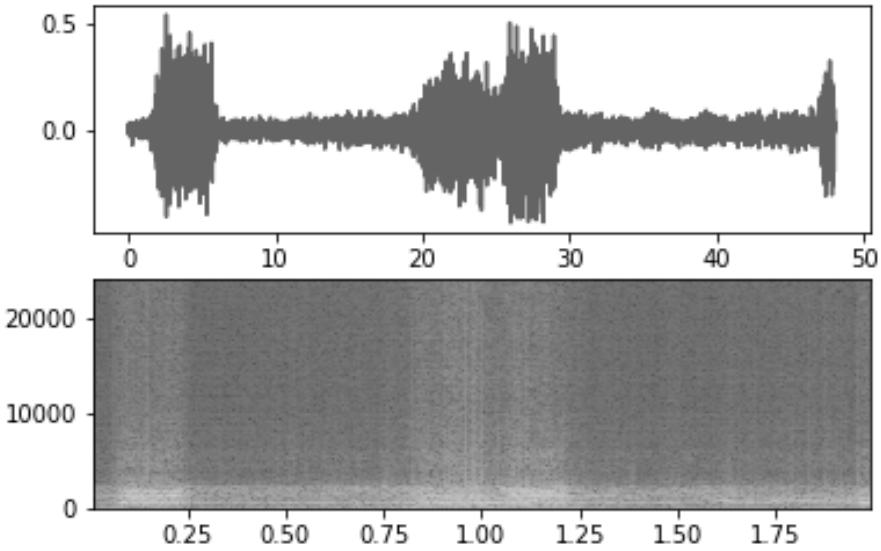
Şekil 4.9: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



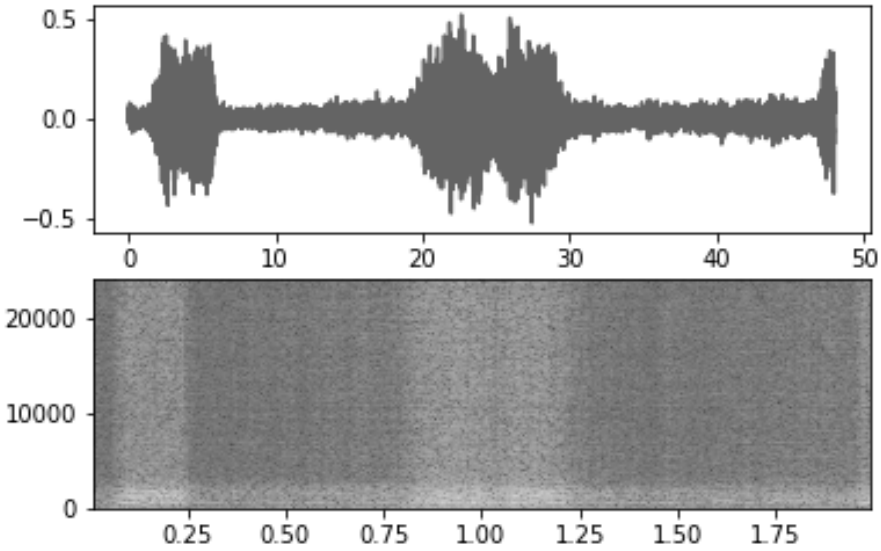
Şekil 4.10: Balon modelinde çalıkuşundan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



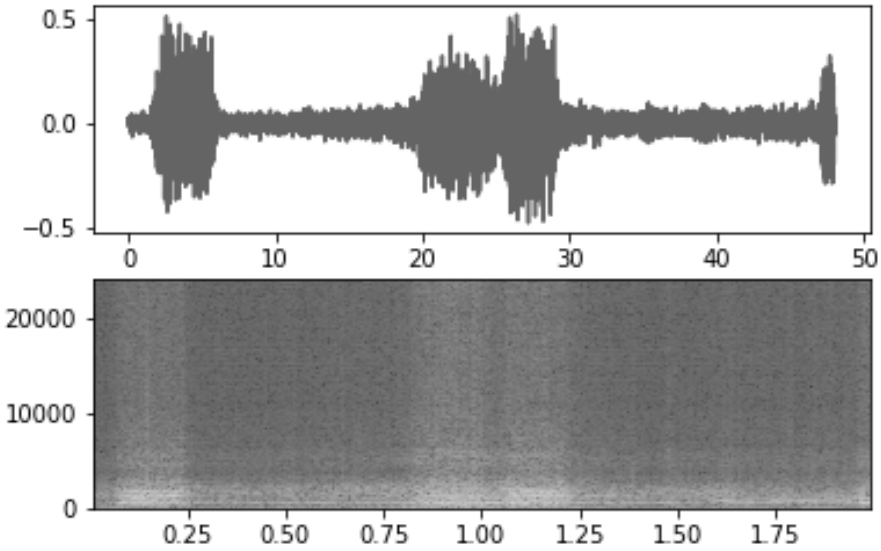
Şekil 4.11: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



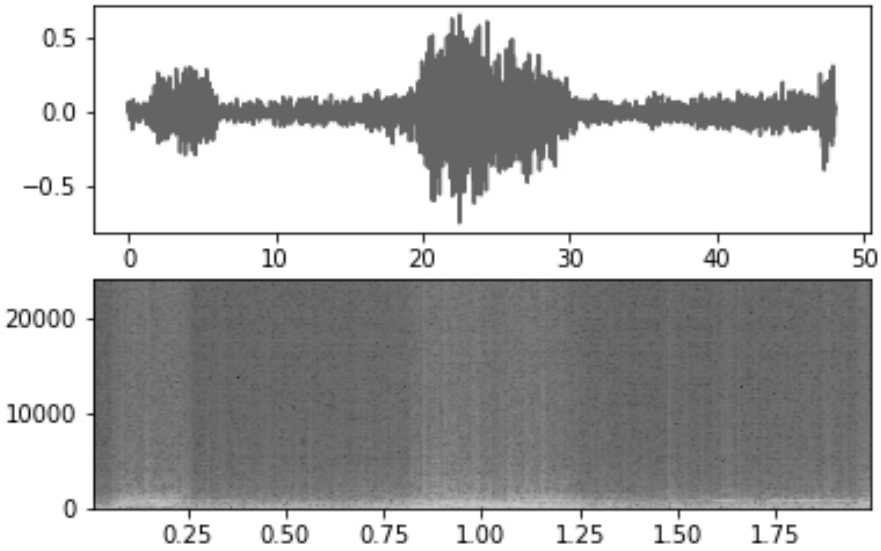
Şekil 4.12: Papyon modelinde kaplandan ayrılmış kangal sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



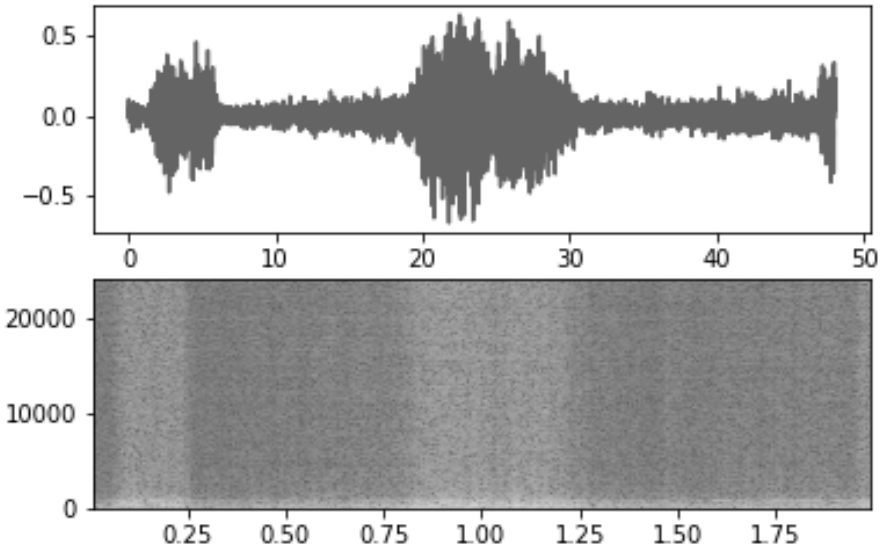
Şekil 4.13: Balon modelinde kaplandan ayrılmış kangal sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



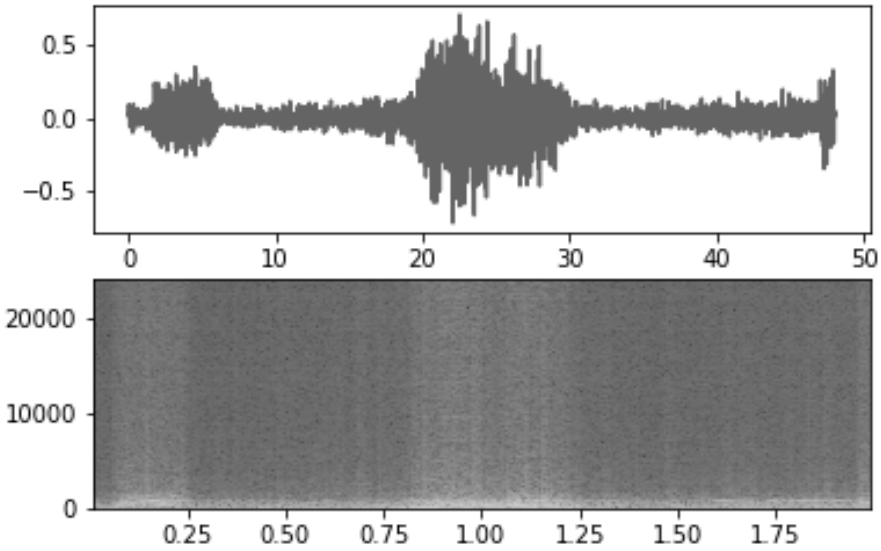
Şekil 4.14: Kova modelinde kaplandan ayrılmış kangal sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 4.15: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 4.16: Balon modelinde kangaldan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 4.17: Kova modelinde kangaldan ayrılmış kaplan sesinin, 120 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.

5 Sonuç ve Değerlendirme

Sonuç olarak önerdiğimiz mimarinin, otokodlayıcı mimarisinin avantajlarından istifade ederek ağımızın öğrenmesini güçlendirdiğini ve performansını iyileştirdiğini gördük. Ağımızın öne çıktığı senaryolar; verinin kısıtlı olduğu ve veri kümelerinin iç içe olduğu, dolayısıyla ayırt etmek için veri içindeki sınıflara ait güçlü öznetliklerin iyi seçilip kullanılması gereken senaryolar oldu. Bunları boyut indirgemesi sayesinde daha performanslı ve verimli şekilde gerçekleştirebildi. Dezavantajlı olduğu durum da verinin görece geniş bir değer aralığında dağılım gösterdiği, fakat kümelenme olan bölgelerde kümelenme yoğunluklarının farklı olduğu bir senaryoydu. Bu durumda yoğunluğu yüksek olan bölgeye ve bunun bir sonucu olarak, ilgili kümede bulunan öznetliklere odaklanma eğilimi gösterdi.

Çalışmamızda karşıaştığımız en büyük sorun, eğitime dönük, açık ve zengin bir ses veritabanı bulamayışımız oldu. Bu durum çalıştığımız veri setlerinin de çok sınırlı uzunluklarda olmasına yol açtı. Denetimli öğrenme yöntemi ile, özellikle büyük ölçekli ve geniş frekans aralıklarında çalışması istenen ağlar üzerinde çalışılabilmesi adına, resim işleme eğitimi için hazırlanmış olan ImageNet'in, ses işleme eğitimindeki muadili sayılabilecek bir büyük ses veritabanına ihtiyaç olduğu açıktır.

Daha önce yapılmış çalışmalara bakıldığında, derin öğrenme eğitimi için, resim işleme alanında kullanılan yaklaşım ve metodolojinin ses işleme alanına uyarlanmasının, ses işleme tarafında ciddi gelişim potansiyeli sunduğu görülüyor. PSNR bize test sonuçlarının başarımlarını sayısal cinsten veriyor. Fakat testi algısal yönden ne derece bir benzerlik, yani başarım gösterdiğine dair bir ölçü sunmuyor. Dolayısıyla bu hususta sesleri yapısal olarak kıyaslayacak bir ölçüye de ihtiyaç olduğu görülüyor. Gelecek çalışmalarda bu ölçü mekanizmalarının geliştirilebileceğini, daha geniş kapsamlı ve büyük ses veritabanlarının oluşturulabileceğini düşünüyoruz. Ve tıpkı resim işleme eğitimlerinde olduğu gibi, bu gelişmelerin güdümünde ses işleme eğitimi için kurulan ağ mimarilerinin ve eğitim süreçlerinin ciddi oranda gelişeceğini öngörüyoruz.

6 Ekler

```
from keras.models import Sequential, Model
from keras import layers
from keras.layers import Input, Dense, Activation, Flatten, Dropout
from keras import optimizers

import soundfile as sf
import numpy as np
import tensorflow as tf

import IPython
import io
import math
```

Kod 6.1: Gerekli kütüphanelerimiz ve alt sınıfları import komutu ile programımıza eklenmiştir.

```

#Ornekler (veri,ornekleme frekansi)

#Calikusu
Cdata,Crate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ FW01.wav’’)

#Kaplan
Edata,Erate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ Tiger01.wav’’)

#Karisim
CEdata,CErate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ FWTiger0101.wav’’)
```

```
12
13
14 #Numpy dizisine çeviri
15 Cdata  = np.array(Cdata)
16 Edata  = np.array(Edata)
17 CEdata = np.array(CEdata)
```

Kod 6.2: Çalıkuşu, kaplan ve karışımlarından oluşan ses üçlümüz kod içinde harf kodlamasıyla ayrılmış ve yönetilmiştir (C ve E harfleri ile). Bu kod bölümünde soundfile.read (sf.read) ile wav ses dosyalarımız okunarak veri dizileri ve frekansları elde edilmiştir. Sonrasında veri dizimize programın devamında kullanılmak üzere, numpy.array (np.array) ile numpy dizisine dönüşüm işlemi uygulanmıştır.

```
1
2 #Ornekler (veri,ornekleme frekansi)
3
4 #Kangal
5 Cdata,Crate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
      ↪ Kangal00.wav’’)
6
7 #Kaplan
8 Edata,Erate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
      ↪ Tiger01.wav’’)
9
10 #Karisim
11 CEdata,CErate = sf.read(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/
      ↪ KangalTiger0001.wav’’)
12
13
14 #Numpy dizisine çeviri
15 Cdata  = np.array(Cdata)
16 Edata  = np.array(Edata)
17 CEdata = np.array(CEdata)
```

Kod 6.3: Kangal, kaplan ve karışımlarından oluşan ses üçlümüz kod içinde harf kodlamasıyla ayrılmış ve yönetilmiştir (C ve E harfleri ile). Bu kod bölümünde soundfile.read (sf.read) ile wav ses dosyalarımız okunarak veri dizileri ve frekansları elde edilmiştir. Sonrasında veri dizimize programın devamında kullanılmak üzere, numpy.array (np.array) ile numpy dizisine dönüşüm işlemi uygulanmıştır.

```
#Tam Veri Uzunlugu: 816000
DATA_DIM      = int(720000)       #%75 (12,75 sn) egitim,
                                  #~%13,24 (2,25 sn) dogrulama
TEST_DATA_DIM = int(96000)  #~%11,76 (2 sn) test

VECTOR_DIM  = int(500)
INPUT_DIM     = int(DATA_DIM / VECTOR_DIM)
TEST_DIM      = int(TEST_DATA_DIM / VECTOR_DIM)


INPUT = np.zeros(DATA_DIM)
OUTPUT = np.zeros(2*DATA_DIM)
TEST = np.zeros(TEST_DATA_DIM)

#Verilerin tek kanal olarak dizilere toplanmasi
for i in range (0,DATA_DIM):
    INPUT[i] = CEdata[i,0]

for i in range (0,DATA_DIM):
    OUTPUT[i] = Cdata[i,0]

for i in range (0,DATA_DIM):
    OUTPUT[i+DATA_DIM] = Edata[i,0]

for i in range (0,TEST_DATA_DIM):
    TEST[i] = CEdata[i+DATA_DIM,0]

#Dizilerin agda kullanilmak uzere iki boyutlu dizi olarak duzenlenmesi
INPUT = INPUT.reshape(INPUT_DIM,VECTOR_DIM)
TEST  = TEST.reshape(TEST_DIM,VECTOR_DIM)

OUT = np.zeros([INPUT_DIM,2*VECTOR_DIM])


for i in range (INPUT_DIM):
  x = OUTPUT[i*VECTOR_DIM:(i+1)*VECTOR_DIM]
  y = OUTPUT[(i+INPUT_DIM)*VECTOR_DIM:(i+INPUT_DIM+1)*
     ↪ VECTOR_DIM]
  OUT[i][:VECTOR_DIM] = x
  OUT[i][VECTOR_DIM:2*VECTOR_DIM] = y
```

Kod 6.4: Veri içinden örnek seçimi için parametrelerimiz bu bölümdedir. DATA_DIM eğitim setinin uzunluğunu, TEST_DATA_DIM ise test setinin uzunluğunu belirlemede kullanılmıştır. Dosyadan okuyup dönüşüm uyguladığımız verilerimiz, eğer ses tek kanallı ise dizi olarak, çok kanallı ise iki boyutlu dizi olarak saklanmaktadır. Bu noktada aynı sesin çift kanalda tutulduğu senaryomuzda sese ait ilk kanalı seçerek tek boyutlu diziye indirgedik. Test için seçtiğimiz karışım test giriş verisini de TEST dizisi ile aldık. Eğitim için giriş verimizi INPUT, çıkış verimizi ise OUTPUT isimlerinde diziler olarak aldık. Bu işlemde çıkış verimiz ayrı haldeki iki kaynağımızın arka arkaya eklenmiş haliyle tek bir dizi olarak alındı. Sonrasında numpy.reshape fonksiyonu kullanarak, INPUT dizimizi vektör boyumuzda (VECTOR_DIM) elemanlardan oluşan bir diziye (iki boyutlu dizi) dönüştürdük. Çıkış katmanımız için aynısını vektör boyunu iki katına çıkararak yaptık ve OUT dizisinde sakladık. OUT, vektör boyumuzun iki katı vektörlerden oluşmaktadır ve ayrı haldeki iki kaynağı sıralı olarak içerir.

```
ins=Input(shape=(VECTOR_DIM,))
x = Dense(int(VECTOR_DIM/2),activation='linear')(ins)
x = Dense(int(VECTOR_DIM/5),activation='linear')(x)
x = Dense(int(VECTOR_DIM/10),activation='linear')(x)
x = Dense(int(VECTOR_DIM/5),activation='linear')(x)
x = Dense(int(VECTOR_DIM/2),activation='linear')(x)
x = Dense(VECTOR_DIM,activation='linear')(x)
out=Dense(2*VECTOR_DIM,activation='linear')(x)

papyon = Model(inputs=ins,outputs=out)

papyon.compile(optimizer='adam' , loss='mean_squared_error')
```

Kod 6.5: Papyon modelimizi, Keras kütüphanesine ait Input, Dense ve Model sınıflarıyla oluşturup Model.compile fonksiyonu ile derledik. Modelimizde optimizatörümüzü Adam optimizatörü, kaybımızı(loss) ise MSE olarak seçtik.

```
ins = Input(shape=(VECTOR_DIM,))
x = Dense(VECTOR_DIM*2,  activation='linear')(ins)
x = Dense(VECTOR_DIM*5,  activation='linear')(x)
x = Dense(VECTOR_DIM*10, activation='linear')(x)
x = Dense(VECTOR_DIM*5,  activation='linear')(x)
x = Dense(VECTOR_DIM*2,  activation='linear')(x)
out = Dense(2*VECTOR_DIM,  activation='linear')(x)

balon = Model(inputs=ins,outputs=out)
```

```
10
11 balon.compile(optimizer='adam' , loss='mean_squared_error')
```

Kod 6.6: Balon modelimizi, Keras kütüphanesine ait Input, Dense ve Model sınıflarıyla oluşturup Model.compile fonksiyonu ile derledik. Modelimizde optimizatörümüzü Adam optimizatörü, kaybımızı(loss) ise MSE olarak seçtik.

```
1 ins = Input(shape=(VECTOR_DIM,))
2 x = Dense(int(VECTOR_DIM+  VECTOR_DIM/5), activation='linear')(ins)
3 x = Dense(int(VECTOR_DIM+2*VECTOR_DIM/5), activation='linear')(x)
4 x = Dense(int(VECTOR_DIM+3*VECTOR_DIM/5), activation='linear')(x)
5 x = Dense(int(VECTOR_DIM+4*VECTOR_DIM/5), activation='linear')(x)
6 out=Dense(2*VECTOR_DIM,      activation='linear')(x)
7
8 kova = Model(inputs=ins,outputs=out)
9
10 kova.compile(optimizer='adam' , loss='mean_squared_error')
```

Kod 6.7: Kova modelimizi, Keras kütüphanesine ait Input, Dense ve Model sınıflarıyla oluşturup Model.compile fonksiyonu ile derledik. Modelimizde optimizatörümüzü Adam optimizatörü, kaybımızı(loss) ise MSE olarak seçtik.

```
1 def psnr(orig, test):
2     mse = np.mean( (orig - test) ** 2 )
3     if mse == 0:
4       return 100
5     PIXEL_MAX = 255.0
6     return 20 * math.log10(np.max(CEdata) / math.sqrt(mse))
```

Kod 6.8: Test sonuçlarımızın istatistiksel tutarlılık derecesini değerlendirmek için kullandığımız PSNR hesabımızı, orjinal veri ve test çıktılarımızı parametre olarak girmek üzere tanımladık [58].

```
1 #Cevrim parametreleri
2 sayi1 = [5,5,5] #5,10,15 çevrim sonuçlari için
3 sayi2 = [15,30,60] #30,60,120 çevrim sonuçlari için
4
```

```
#PSNR kaydi için listeler
Ppsnr = [] #papyon
Bpsnr = [] #balon
Kpsnr = [] #kova

#Sonuçlarin kaydi için listeler
Pouts = [] #papyon
Bouts = [] #balon
Kouts = [] #kova

for s in sayi1: # 3x5 cevrim (model başina 5)
  papyon.fit(INPUT,OUT,epochs=s,batch_size=30, validation_split=0.15)
  balon.fit(INPUT,OUT,epochs=s,batch_size=30, validation_split=0.15)
  kova.fit(INPUT,OUT,epochs=s,batch_size=30, validation_split=0.15)
  # Test tahminleri (papyon)
  Ppredictions = papyon.predict(TEST[:TEST_DIM,:VECTOR_DIM],batch_size=30)
  Ppredictions = np.asarray(Ppredictions)
  PpredictionsC = Ppredictions[:,0:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
  PpredictionsE = Ppredictions[:,VECTOR_DIM:2*VECTOR_DIM].reshape(
    ↪ TEST_DATA_DIM,1)
  #Tahminlerin sentezi (papyon)
  PCSynth = PpredictionsC
  PESynth = PpredictionsE

  #Balon
  Bpredictions = balon.predict(TEST[:TEST_DIM,:VECTOR_DIM],batch_size=30)
  Bpredictions = np.asarray(Bpredictions)
  BpredictionsC = Bpredictions[:,0:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
  BpredictionsE = Bpredictions[:,VECTOR_DIM:2*VECTOR_DIM].reshape(
    ↪ TEST_DATA_DIM,1)
  BCSynth = BpredictionsC
  BESynth = BpredictionsE
  #Kova
  Kpredictions = kova.predict(TEST[:TEST_DIM,:VECTOR_DIM],batch_size=30)
  Kpredictions = np.asarray(Kpredictions)
  KpredictionsC = Kpredictions[:,0:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
  KpredictionsE = Kpredictions[:,VECTOR_DIM:2*VECTOR_DIM].reshape(
    ↪ TEST_DATA_DIM,1)
  KCSynth = KpredictionsC
  KESynth = KpredictionsE

  #PSNR hesaplanmasi ve kaydedilmesi (papyon)
  PCpsnr=psnr(Cdata[DATA_DIM:],PpredictionsC)
```

```
45    PEpsnr=psnr(Edata[DATA_DIM:],PpredictionsE)
46    Ppsnr.append((PCpsnr,PEpsnr))
47
48    BCpsnr=psnr(Cdata[DATA_DIM:],BpredictionsC)
49    BEpsnr=psnr(Edata[DATA_DIM:],BpredictionsE)
50    Bpsnr.append((BCpsnr,BEpsnr))
51
52    KCpsnr=psnr(Cdata[DATA_DIM:],KpredictionsC)
53    KEpsnr=psnr(Edata[DATA_DIM:],KpredictionsE)
54    Kpsnr.append((KCpsnr,KEpsnr))
55
56    Pouts.append(PpredictionsC)
57    Pouts.append(PpredictionsE)
58
59    Bouts.append(BpredictionsC)
60    Bouts.append(BpredictionsE)
61
62    Kouts.append(KpredictionsC)
63    Kouts.append(KpredictionsE)
64
65  for t in sayi2: # 15, 30 ve 60 cevrim
66    papyon.fit(INPUT,OUT,epochs=t,batch_size=30, validation_split=0.15)
67    kova.fit(INPUT,OUT,epochs=t,batch_size=30, validation_split=0.15)
68
69    Ppredictions = papyon.predict(TEST[:TEST_DIM,:VECTOR_DIM],batch_size=30)
70    Ppredictions = np.asarray(Ppredictions)
71    PpredictionsC = Ppredictions[:,0:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
72    PpredictionsE = Ppredictions[:,VECTOR_DIM:2*VECTOR_DIM].reshape(
        ↪ TEST_DATA_DIM,1)
73    PCSynth = PpredictionsC
74    PESynth = PpredictionsE
75
76    Kpredictions = kova.predict(TEST[:TEST_DIM,:VECTOR_DIM],batch_size=30)
77    Kpredictions = np.asarray(Kpredictions)
78    KpredictionsC = Kpredictions[:,0:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
79    KpredictionsE = Kpredictions[:,VECTOR_DIM:2*VECTOR_DIM].reshape(
        ↪ TEST_DATA_DIM,1)
80    KCSynth = KpredictionsC
81    KESynth = KpredictionsE
82
83
84    PCpsnr=psnr(Cdata[DATA_DIM:],PpredictionsC)
85    PEpsnr=psnr(Edata[DATA_DIM:],PpredictionsE)
```

```
86  Ppsnr.append((PCpsnr,PEpsnr))
87
88  KCpsnr=psnr(Cdata[DATA_DIM:],KpredictionsC)
89  KEpsnr=psnr(Edata[DATA_DIM:],KpredictionsE)
90  Kpsnr.append((KCpsnr,KEpsnr))
91
92  Pouts.append(PpredictionsC)
93  Pouts.append(PpredictionsE)
94
95  Kouts.append(KpredictionsC)
96  Kouts.append(KpredictionsE)
```

Kod 6.9: Ağlarımızı Model.fit fonksiyonu ile eğitime tabi tuttuk. İlk iki parametre olan giriş ve çıkış verilerimiz haricinde, çevrim sayımızı (epochs), grup büyüklüğümüzü (batch_size) ve doğrulama ayrım oranımızı (validation_split) girdik. Doğrulama ayrım oranımız aldığı parametre oranında veriyi, giriş ve çıkış verilerinin sonundan ayırarak her çevrim bitiminde bu veriyi modeli sınamak için kullanır. Bu işlem sonucunda doğrulama kaybını verir. Modellerimize ait değişkenler baş harfleri ile kodlandı. Prosedürümzde değişken çevrim sayıları ile (sırası ile sayi1 ve sayi2 içindeki sayılar) 5,10,15,30,60 ve 120 çevrim eğitimler aşamalı olarak gerçekleştirildi. Her eğitim aşamasının bitiminde modellerimize ait PSNR değerleri ve sonuç sentezleri listelere kaydedildi. Test karışım verisinden modelimize ayrıştırdığı kaynakları sentezletmek için Model.predict fonksiyonunu kullandık. Modellerimizin çıktıları, grafik çizimi ve vaw dosyası yazımında kullanılmak üzere tek boyutlu dizi olarak yeniden düzenlendi. Bu süreçte, Balon modelinin 10-15 çevrim sonunda azami kararlılığa ulaşmasından ötürü rutin iki parçaya bölündü ve sayi2 değerleri ile işletilen 30,60 ve 120 çevrim rutininden çıkarıldı.

```
1
2  source1 = [0,2,4,6,8,10]
3  source2 = [1,3,5,7,9,11]
4  source3 = [0,2,4]
5  source4 = [1,3,5]
6
7  epoch1 = [5,10,15,30,60,120]
8  epoch2 = [5,10,15]
9
10 Fs = 48000
11
12 i = 0
13 for s in source1:
```

```
14    a = np.array(Pouts[s][:])
15    x = a[:,0]
16
17    sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Pfw’’ +
          ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)
18
19    i = i + 1
20
21  i = 0
22  for s in source2:
23    a = np.array(Pouts[s][:])
24    x = a[:,0]
25
26    sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/P0t’’ +
          ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)
27
28    i = i + 1
29
30  i = 0
31  for s in source3:
32    a = np.array(Bouts[s][:])
33    x = a[:,0]
34
35    sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Bfw’’ +
          ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)
36
37    i = i + 1
38
39  i = 0
40  for s in source4:
41    a = np.array(Bouts[s][:])
42    x = a[:,0]
43
44    sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/B0t’’ +
          ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)
45
46    i = i + 1
47
48  i = 0
49  for s in source1:
50    a = np.array(Kouts[s][:])
51    x = a[:,0]
52
```

```
  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Kfw’’ +
    ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

i = 0
for s in source2:
  a = np.array(Kouts[s][:])
  x = a[:,0]

  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/K0t’’ +
    ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

CETest = TEST[:TEST_DIM,:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/FWTSynth.wav’’,
    ↪ CETest, 48000)
```

Kod 6.10: Çalıkuşu-kaplan karışımı eğitim verisi üzerindeki test sonuçlarınının wav sentezleri prosedürel olarak üretilmiştir. Sonuç listelerimizde ayrılmış kaynaklarımız birbirinden ayrı olarak çift ve tek indekslerde tutulmaktadır. Her döngümüzde bir modelden bir kaynağa ait sonuçlar sırasıyla seçilir ve diziye dönüştürüldükten sonra soundfile.write (sf.write) ile wav dosyasına yazılır. Parametreler olarak dosya yoluyla birlikte dosya adı, veri dizisi ve örnekleme frekansı verilir. Dosya ismi harf kodlamamızda fw (fulvous wren) çalıkuşunu, k (kangal) kangal köpeğini, t (tiger) ise kaplanı temsil etmektedir. Kaplan örneğimiz iki karışımda da bulunduğu için çalıkuşundan ayrılmış kaplan ve kangaldan ayrılmış kaplan örnekleri için sırasıyla 0t ve 1t isim kodlaması yapılmıştır. Bu isimlerden sonra ise ilgili sonucun çevrim sayısı bulunur. Son olarak seçtiğimiz test verisi de ayrıca dosyaya yazılır.

```

source1 = [0,2,4,6,8,10]
source2 = [1,3,5,7,9,11]
source3 = [0,2,4]
source4 = [1,3,5]

epoch1 = [5,10,15,30,60,120]
epoch2 = [5,10,15]

Fs = 48000

```

```
i = 0
for s in source1:
  a = np.array(Pouts[s][:])
  x = a[:,0]

  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Pk’’ + str
    ↪ (epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

i = 0
for s in source2:
  a = np.array(Pouts[s][:])
  x = a[:,0]

  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/P1t’’ +
    ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

i = 0
for s in source3:
  a = np.array(Bouts[s][:])
  x = a[:,0]

  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Bk’’ + str
    ↪ (epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

i = 0
for s in source4:
  a = np.array(Bouts[s][:])
  x = a[:,0]

  sf.write(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/B1t’’ +
    ↪ str(epoch1[i]) + ‘‘.wav’’, x, Fs)

  i = i + 1

i = 0
for s in source1:
  a = np.array(Kouts[s][:])
```

```
  x = a[:,0]

  sf.write(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Kk'' +
      ↪ str(epoch1[i]) + ''.wav'', x, Fs)

  i = i + 1

i = 0
for s in source2:
  a = np.array(Kouts[s][:])
  x = a[:,0]

  sf.write(''/content/drive/My Drive/Colab/SourceSeperation/tested results/K1t'' +
      ↪ str(epoch1[i]) + ''.wav'', x, Fs)

  i = i + 1

CETest = TEST[:TEST_DIM,:VECTOR_DIM].reshape(TEST_DATA_DIM,1)
sf.write(''/content/drive/My Drive/Colab/SourceSeperation/data/KTSynth.wav'',
    ↪ CETest, 48000)
```

Kod 6.11: Kangal-kaplan karışımı eğitim verisi üzerindeki test sonuçlarınının wav sentezleri prosedürel olarak üretilmiştir. Sonuç listelerimizde ayrılmış kaynaklarımız birbirinden ayrı olarak çift ve tek indekslerde tutulmaktadır. Her döngümüzde bir modelden bir kaynağa ait sonuçlar sırasıyla seçilir ve diziye dönüştürüldükten sonra soundfile.write (sf.write) ile wav dosyasına yazılır. Parametreler olarak dosya yoluyla birlikte dosya adı, veri dizisi ve örnekleme frekansı verilir. Dosya ismi harf kodlamamızda fw (fulvous wren) çalıkuşunu, k (kangal) kangal köpeğini, t (tiger) ise kaplanı temsil etmektedir. Kaplan örneğimiz iki karışımda da bulunduğu için çalıkuşundan ayrılmış kaplan ve kangaldan ayrılmış kaplan örnekleri için sırasıyla 0t ve 1t isim kodlaması yapılmıştır. Bu isimlerden sonra ise ilgili sonucun çevrim sayısı bulunur. Son olarak seçtiğimiz test verisi de ayrıca dosyaya yazılır.

```
#Calisma araligi
dt = 0.0005
t = np.arange(0.0, 48.0, dt)

NFFT = 1000
Fs = 48000

#prosedurel parametreler
source1 = [0,2,4,6,8,10]
```

```
source2 = [1,3,5,7,9,11]
source3 = [0,2,4]
source4 = [1,3,5]
epoch1 = [5,10,15,30,60,120]
epoch2 = [5,10,15]

#Papyon sonuclari
i = 0
for s in source1:
  a = np.array(Pouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Pfw'' +
    ↪ str(epoch1[i]) + ''.png'')
  i = i + 1

i = 0
for s in source2:
  a = np.array(Pouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/P0t'' +
    ↪ str(epoch1[i]) + ''.png'')
  i = i + 1

#Balon sonuclari
i = 0
for s in source3:
  a = np.array(Bouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Bfw'' +
    ↪ str(epoch2[i]) + ''.png'')
  i = i + 1
```

```
50
51 i = 0
52 for s in source4:
53   a = np.array(Bouts[s][:])
54   x = a[:,0]
55
56   fig, (ax1, ax2) = plt.subplots(nrows=2)
57   ax1.plot(t, x)
58   Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
59   fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/B0t'' +
       ↪ str(epoch2[i]) + ''.png'')
60   i = i + 1
61
62 #Kova sonuclari
63 i = 0
64 for s in source1:
65   a = np.array(Kouts[s][:])
66   x = a[:,0]
67
68   fig, (ax1, ax2) = plt.subplots(nrows=2)
69   ax1.plot(t, x)
70   Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
71   fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Kfw''
       ↪ + str(epoch1[i]) + ''.png'')
72   i = i + 1
73
74 i = 0
75 for s in source2:
76   a = np.array(Kouts[s][:])
77   x = a[:,0]
78
79   fig, (ax1, ax2) = plt.subplots(nrows=2)
80   ax1.plot(t, x)
81   Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
82   fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/K0t'' +
       ↪ str(epoch1[i]) + ''.png'')
83   i = i + 1
```

Kod 6.12: Kod 6.10 ve 6.11' de kullanılan prosedür ve isim kodlama uyarlanarak çalıkuşu-kaplan karışımı test sonuçlarının genlik-zaman ve spektrogram grafikleri prosedürel olarak çizdirilmiş ve dosyaya kaydedilmiştir [59, 60]. Grafik çizdirme için matplotlib kütühanesi ve kütüphaneye ait pyplot ile axis sınıfları kullanılmıştır. Genlik-zaman grafiği için plot fonksiyonundan, spektrogram için ise specgram fonksiyonundan faydalanılmıştır.

```
#Calisma araligi
dt = 0.0005
t = np.arange(0.0, 48.0, dt)

#FFT penceresi
NFFT = 1000

#Ornekleme frekansi
Fs = 48000

#Prosedurel parametreler
source1 = [0,2,4,6,8,10]
source2 = [1,3,5,7,9,11]
source3 = [0,2,4]
source4 = [1,3,5]
epoch1 = [5,10,15,30,60,120]
epoch2 = [5,10,15]

#Papyon sonuclari

#Kangal
i = 0
for s in source1:
  a = np.array(Pouts[s][:])
  x = a[:,0]
#Grafikler
  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(‘‘/content/drive/My Drive/Colab/SourceSeperation/tested results/Pk’’ +
     ↪ str(epoch1[i]) + ‘‘.png’’)
  i = i + 1

#Kaplan
i = 0
for s in source2:
  a = np.array(Pouts[s][:])
  x = a[:,0]
#Grafikler
  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
```

```
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/P1t'' + str(epoch1[i]) + ''.png'')
  i = i + 1

#Balon sonuclari

#Kangal
i = 0
for s in source3:
  a = np.array(Bouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Bk'' + str(epoch2[i]) + ''.png'')
  i = i + 1

#Kaplan
i = 0
for s in source4:
  a = np.array(Bouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/B1t'' + str(epoch2[i]) + ''.png'')
  i = i + 1

#Kova sonuclari

#Kangal
i = 0
for s in source1:
  a = np.array(Kouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
```

```
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/Kk'' +
    ↪ str(epoch1[i]) + ''.png'')
  i = i + 1

#Kaplan
i = 0
for s in source2:
  a = np.array(Kouts[s][:])
  x = a[:,0]

  fig, (ax1, ax2) = plt.subplots(nrows=2)
  ax1.plot(t, x)
  Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
  fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/tested results/K1t'' +
    ↪  str(epoch1[i]) + ''.png'')
  i = i + 1
```

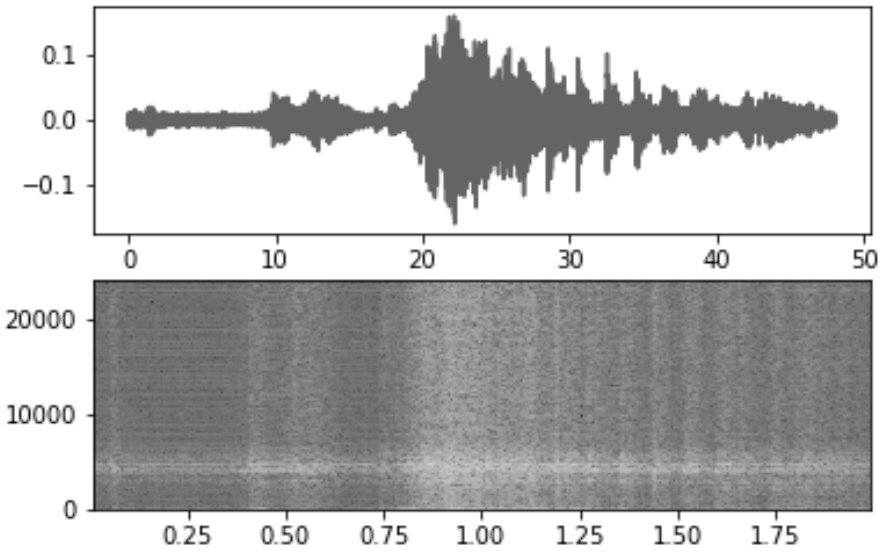
Kod 6.13: Kod 6.10 ve 6.11' de kullanılan prosedür ve isim kodlama uyarlanarak kangal-kaplan karışımı test sonuçlarının genlik-zaman ve spektrogram grafikleri prosedürel olarak çizdirilmiş ve dosyaya kaydedilmiştir [59, 60]. Grafik çizdirme için matplotlib kütühanesi ve kütüphaneye ait pyplot ile axis sınıfları kullanılmıştır. Genlik-zaman grafiği için plot fonksiyonundan, spektrogram için ise specgram fonksiyonundan faydalanılmıştır.

```
#Ornekler
Adata,Arate = sf.read(''/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ FW01.wav'')
Bdata,Brate = sf.read(''/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ Tiger01.wav'')
Cdata,Crate = sf.read(''/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ Kangal00.wav'')
#Karisimlar
ABdata,ABrate = sf.read(''/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ FWTiger0101.wav'')
BCdata,BCrate = sf.read(''/content/drive/My Drive/Colab/SourceSeperation/data/
    ↪ KangalTiger0001.wav'')

#Dizilerin numpy d"on"Us"Um"U
Adata  = np.array(Adata)
Bdata  = np.array(Bdata)
Cdata  = np.array(Cdata)
ABdata = np.array(ABdata)
```

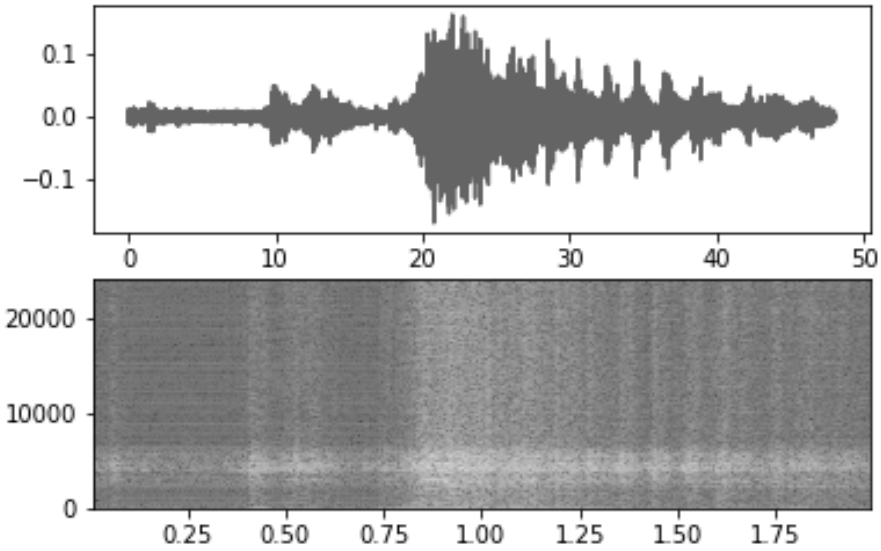
```
BCdata = np.array(BCdata)

#Cizim araligi (t) ve adim buyuklugu (dt)
dt = 0.0005
t = np.arange(0.0, 48.0, dt)

w = ABdata[720000:816000,0]
v = BCdata[720000:816000,0]
x =  Adata[720000:816000,0]
y =  Bdata[720000:816000,0]
z =  Cdata[720000:816000,0]

NFFT = 1000 #FFT pencere buyuklugu
Fs = 48000 #Ornekleme frekansi


fig, (ax1, ax2) = plt.subplots(nrows=2) #Grafik cizim alanlarinin tanimlanmasi
ax1.plot(t, x) #Genlik−zaman grafigi
Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900) #
    ↪ Spektrogram
fig.savefig(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/fw.png’’) #Cizimi
    ↪  kaydet

fig, (ax1, ax2) = plt.subplots(nrows=2) #Grafik cizim alanlarinin tanimlanmasi
ax1.plot(t, y) #Genlik−zaman grafigi
Pxx, freqs, bins, im = ax2.specgram(y, NFFT=NFFT, Fs=Fs, noverlap=900) #
    ↪ Spektrogram
fig.savefig(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/t.png’’) #Cizimi
    ↪ kaydet

fig, (ax1, ax2) = plt.subplots(nrows=2) #Grafik cizim alanlarinin tanimlanmasi
ax1.plot(t, z) #Genlik−zaman grafigi
Pxx, freqs, bins, im = ax2.specgram(z, NFFT=NFFT, Fs=Fs, noverlap=900) #
    ↪ Spektrogram
fig.savefig(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/k.png’’) #Cizimi
    ↪ kaydet

fig, (ax1, ax2) = plt.subplots(nrows=2) #Grafik cizim alanlarinin tanimlanmasi
ax1.plot(t, w) #Genlik−zaman grafigi
Pxx, freqs, bins, im = ax2.specgram(w, NFFT=NFFT, Fs=Fs, noverlap=900) #
    ↪ Spektrogram
fig.savefig(‘‘/content/drive/My Drive/Colab/SourceSeperation/data/fwt.png’’) #
    ↪ Cizimi kaydet
```

```
49
50 fig, (ax1, ax2) = plt.subplots(nrows=2) #Grafik cizim alanlarinin tanimlanmasi
51 ax1.plot(t, v) #Genlik−zaman grafigi
52 Pxx, freqs, bins, im = ax2.specgram(v, NFFT=NFFT, Fs=Fs, noverlap=900) #
       ↪ Spektrogram
53 fig.savefig(''/content/drive/My Drive/Colab/SourceSeperation/data/kt.png'') #Cizimi
       ↪ kaydet
```

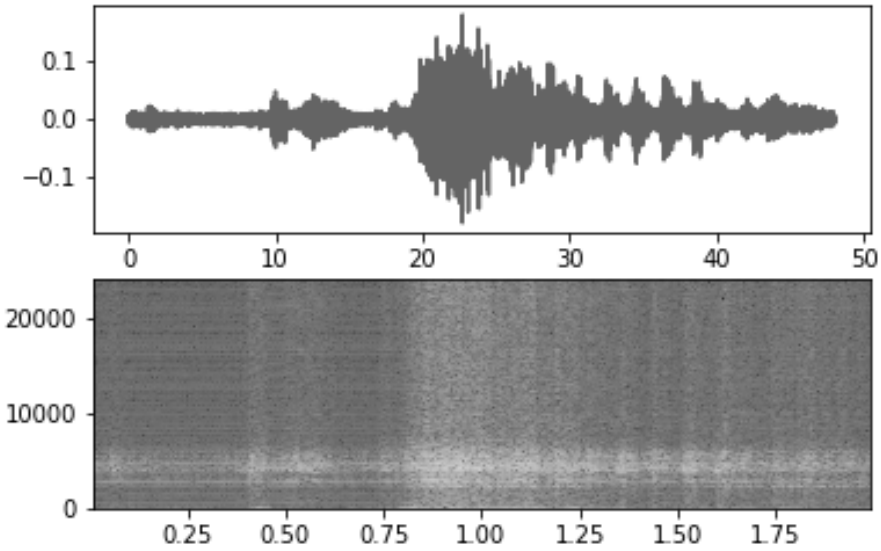
Kod 6.14: 6.12 ve 6.13' teki fonksiyonlar parametrik şekilde kullanılarak test setlerinin genlik-zaman grafikleri ve spektrogramları çizdirilmiş ve dosyaya kaydedilmiştir.



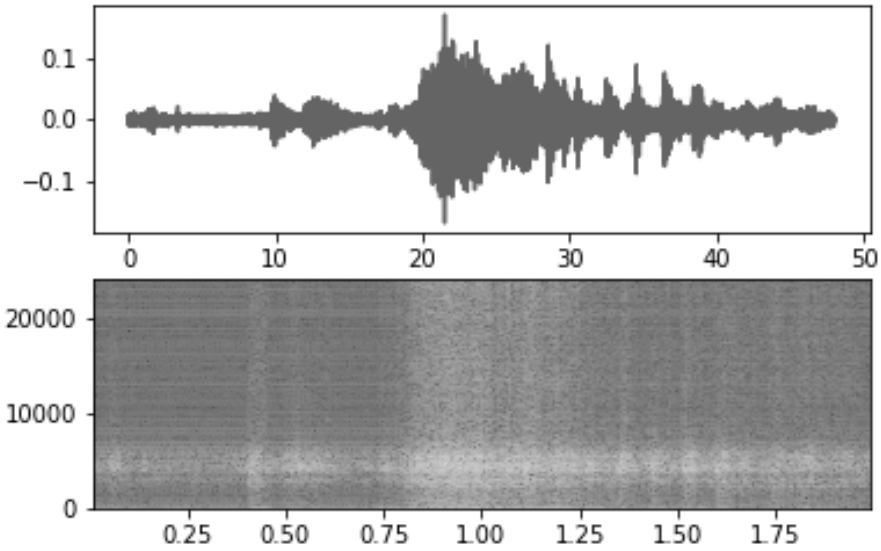
Şekil 6.1: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



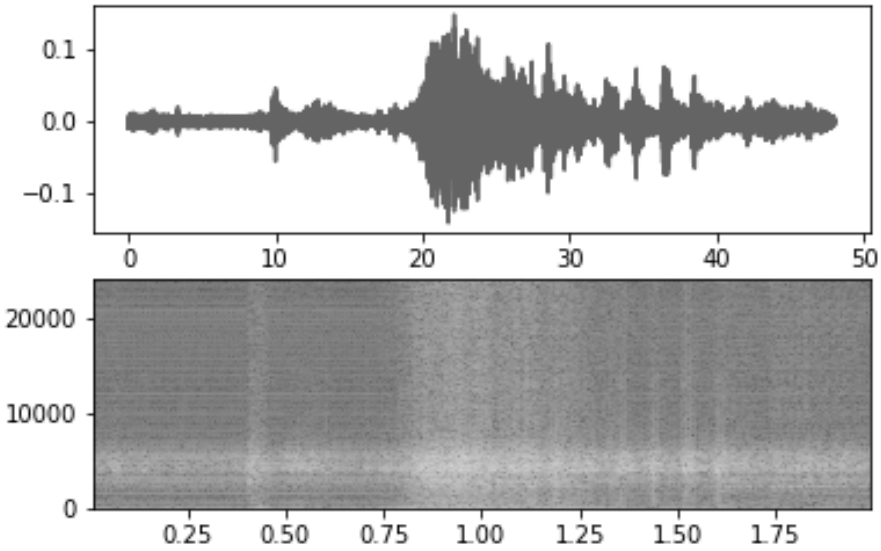
Şekil 6.2: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



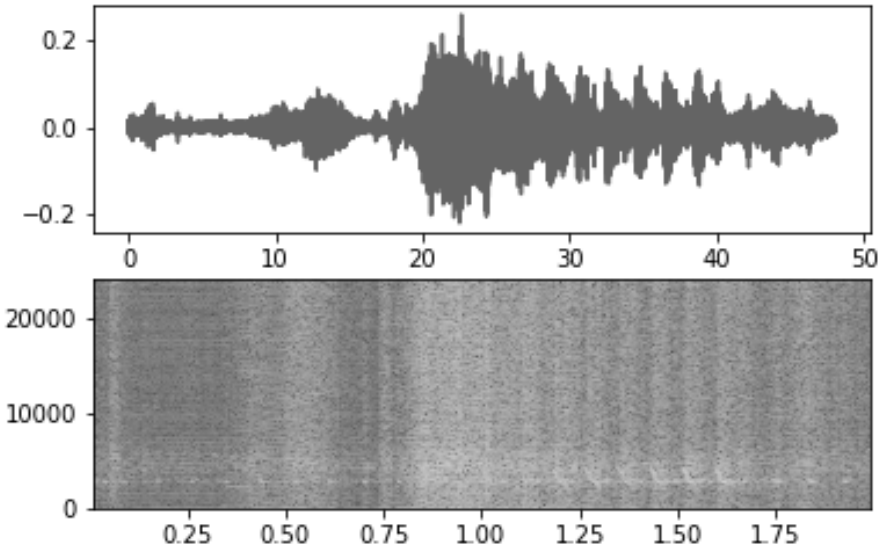
Şekil 6.3: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



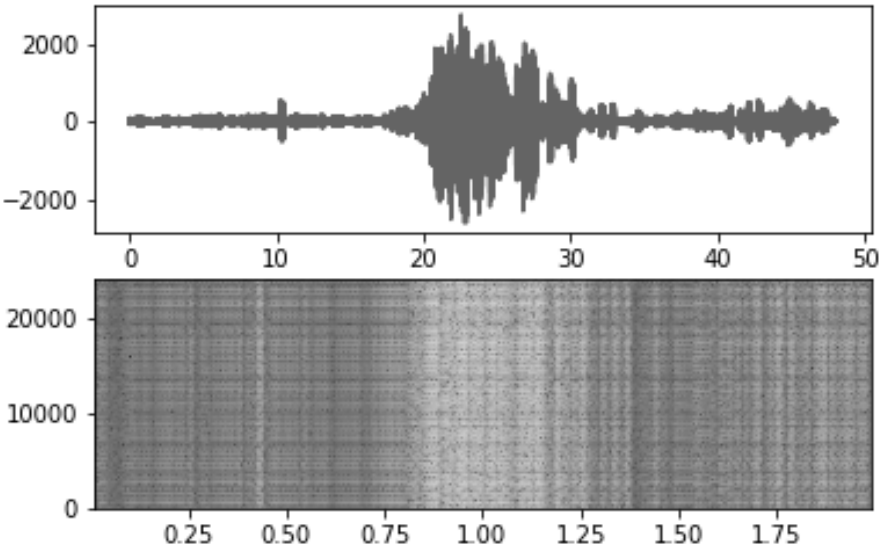
Şekil 6.4: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



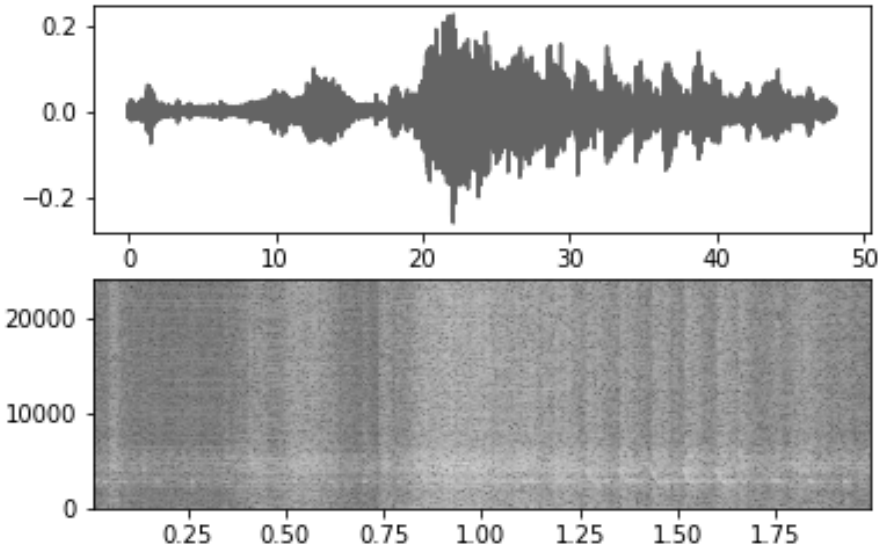
Şekil 6.5: Papyon modelinde kaplandan ayrılmış çalıkuşu sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



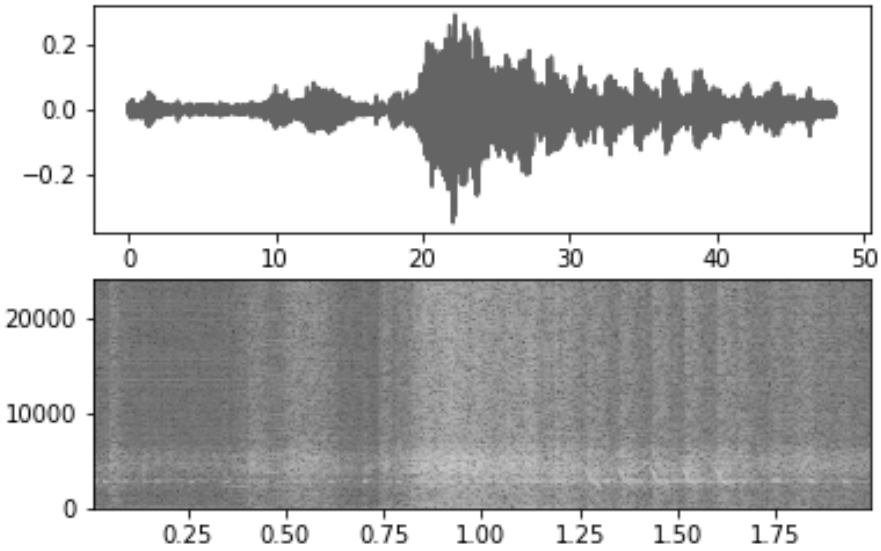
Şekil 6.6: Balon modelinde kaplandan ayrılmış çalıkuşu sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



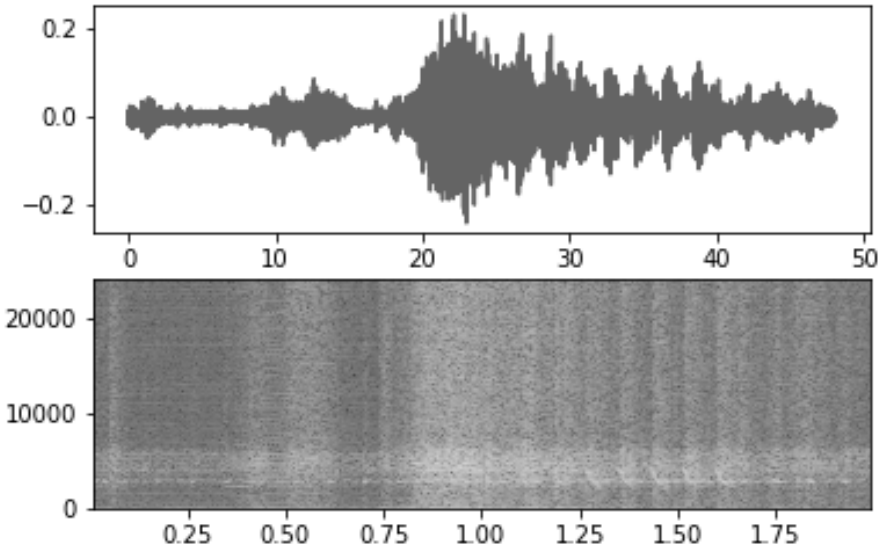
Şekil 6.7: modelinde kaplandan ayrılmış çalıkuşu sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



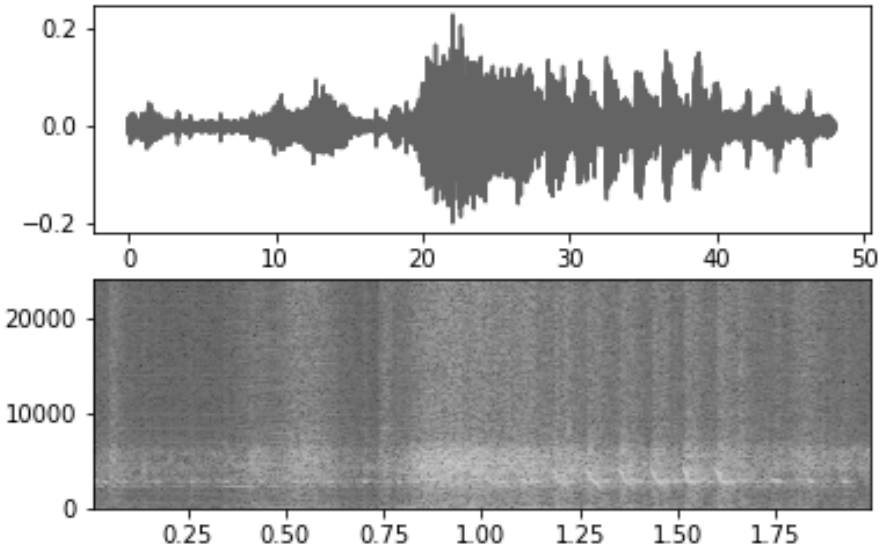
Şekil 6.8: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



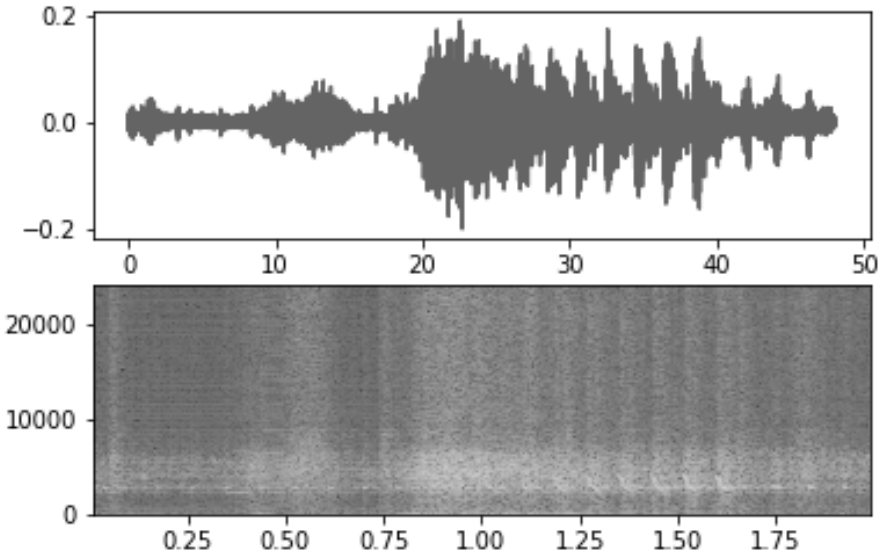
Şekil 6.9: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



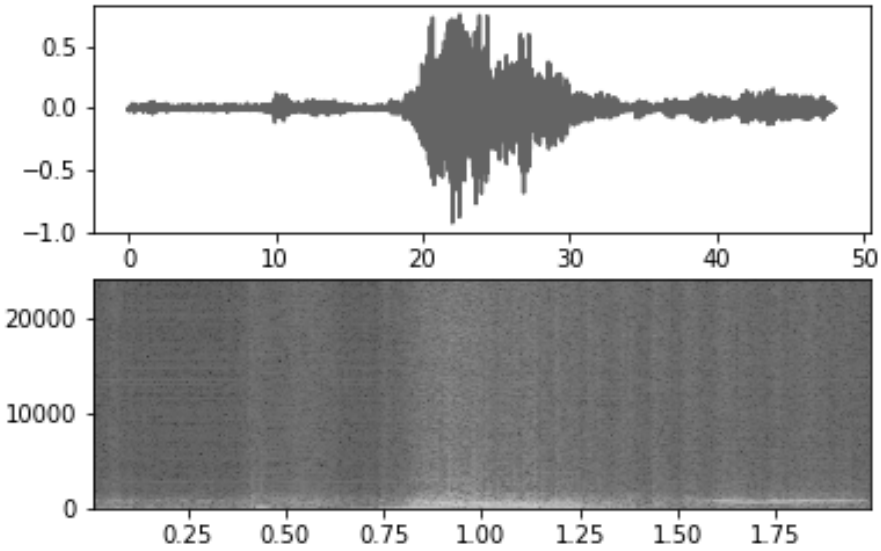
Şekil 6.10: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



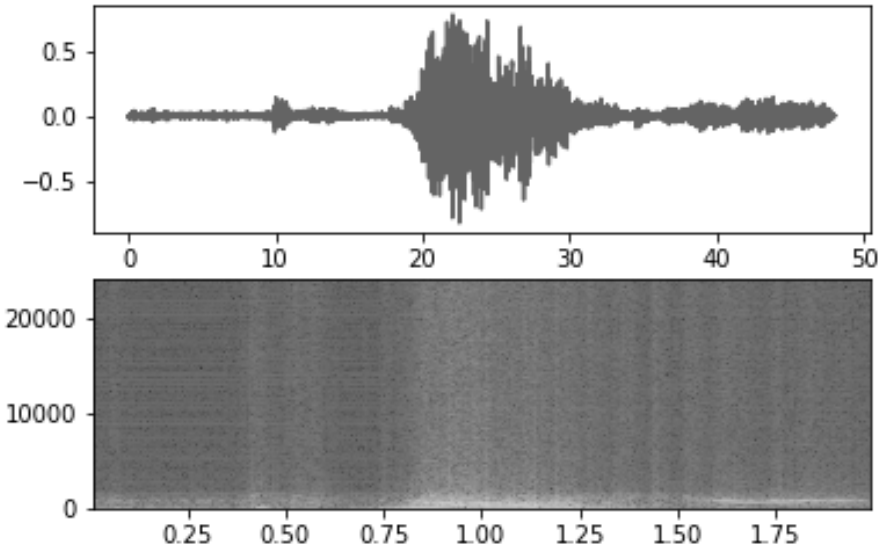
Şekil 6.11: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



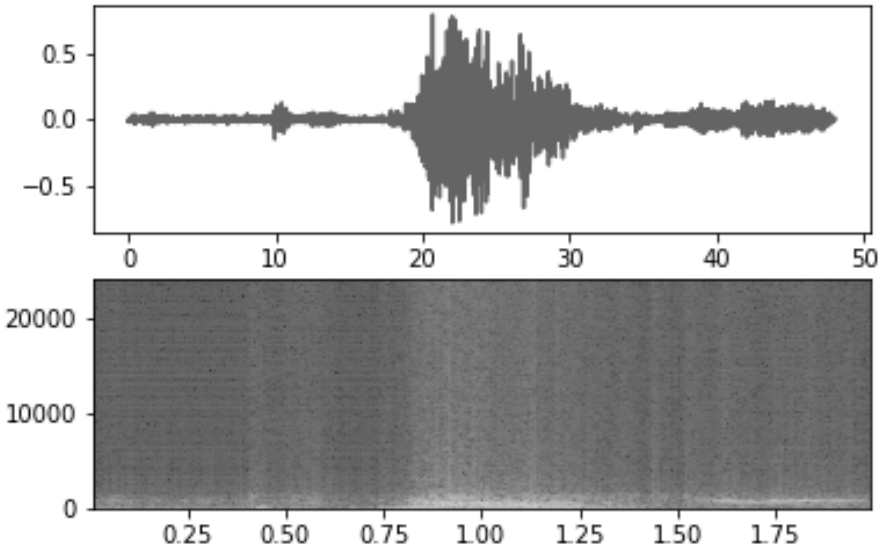
Şekil 6.12: Kova modelinde kaplandan ayrılmış çalıkuşu sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



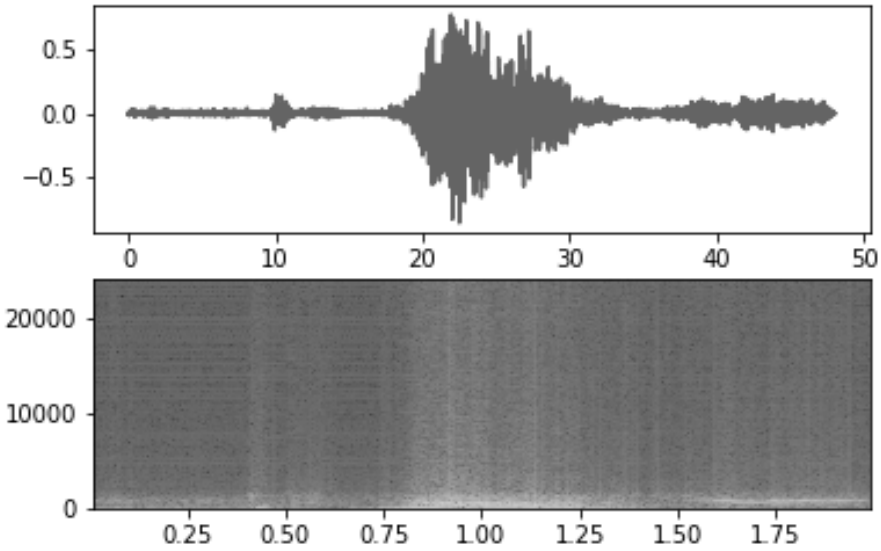
Şekil 6.13: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



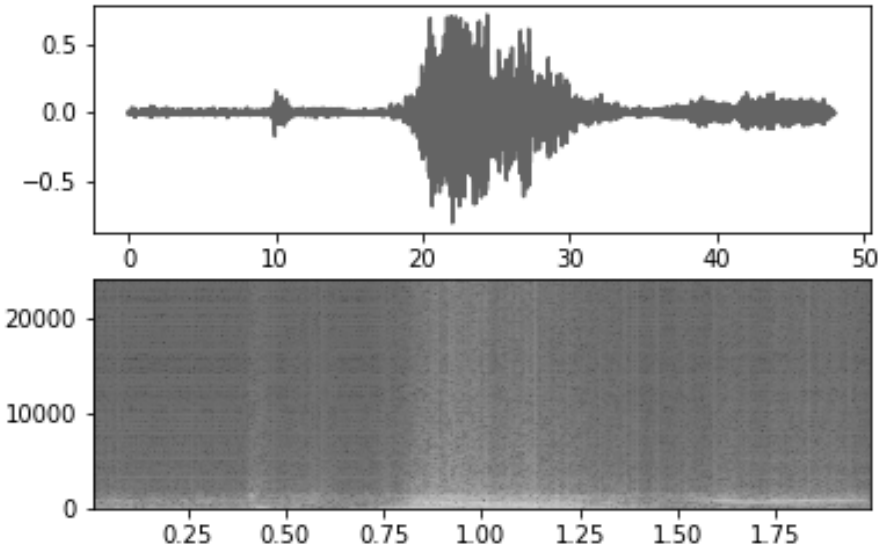
Şekil 6.14: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



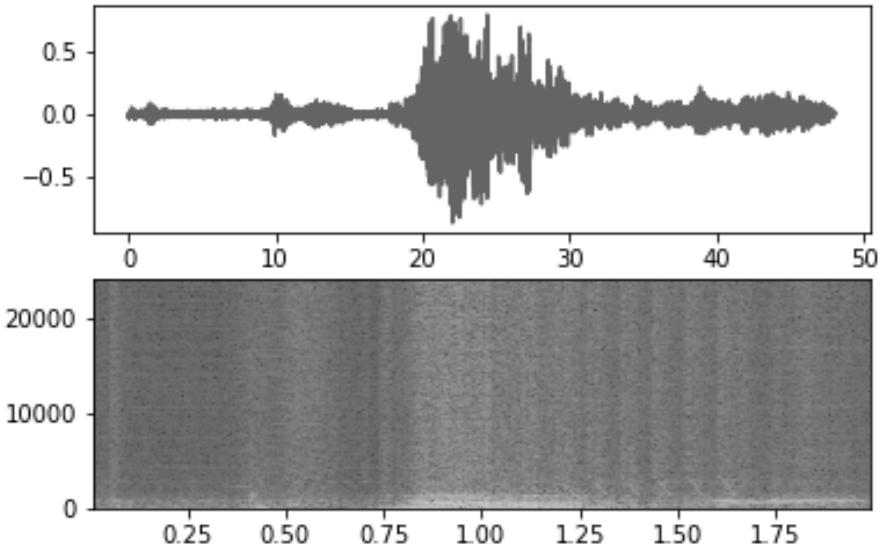
Şekil 6.15: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



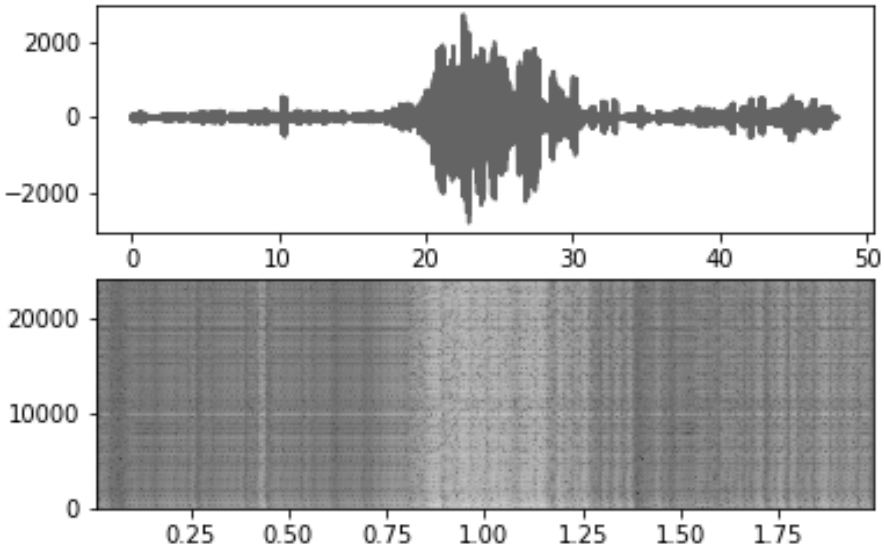
Şekil 6.16: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



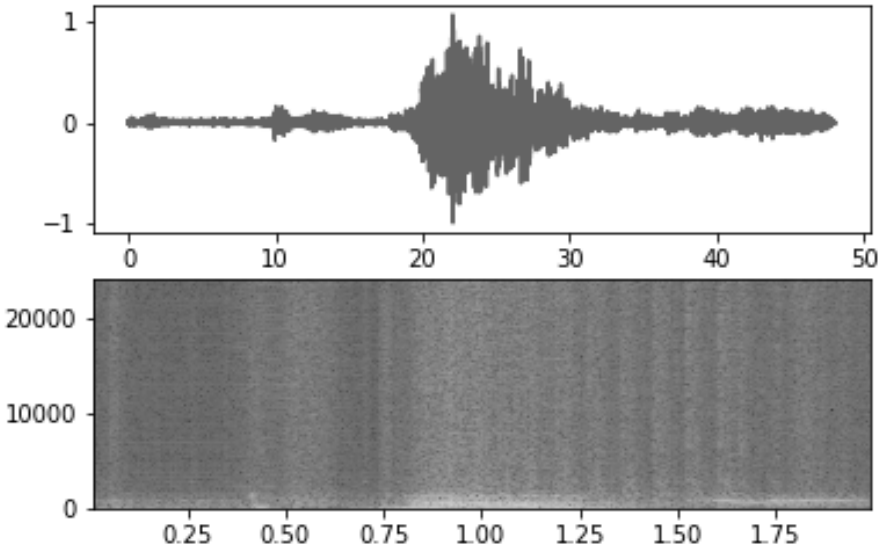
Şekil 6.17: Papyon modelinde çalıkuşundan ayrılmış kaplan sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



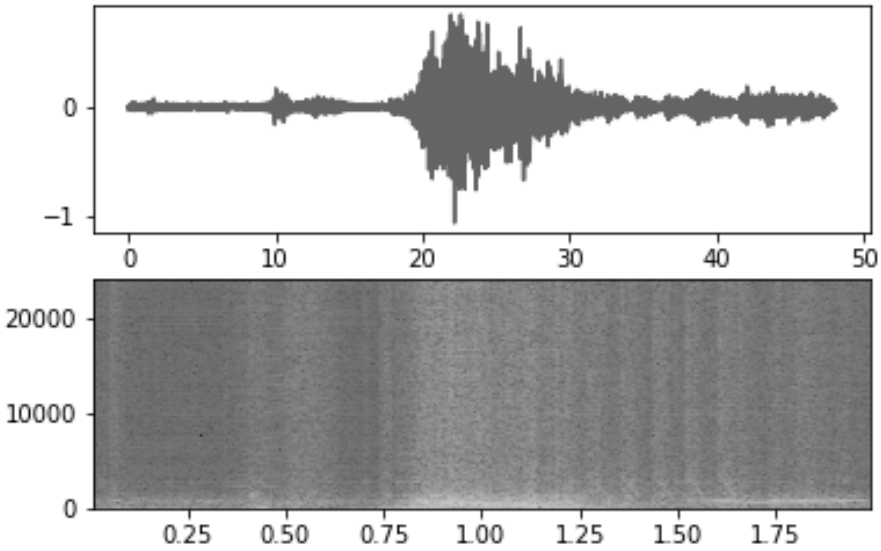
Şekil 6.18: Balon modelinde çalıkuşundan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



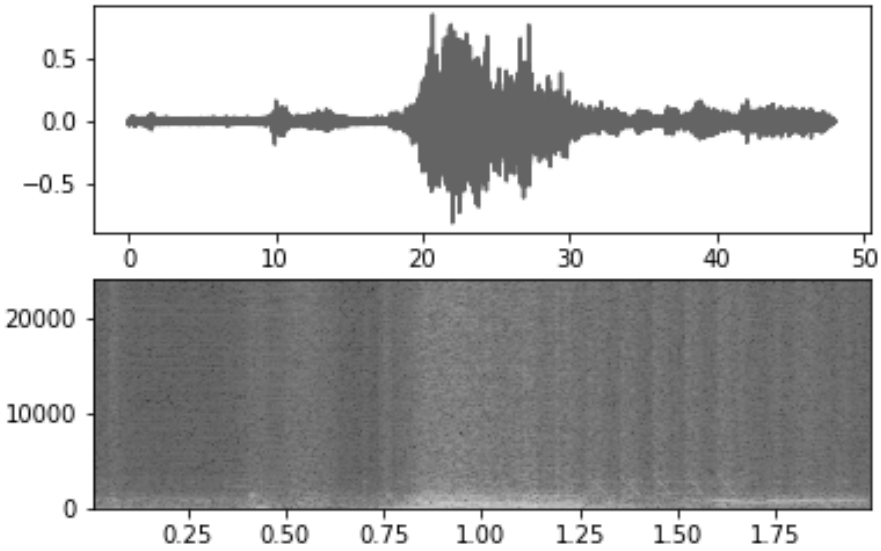
Şekil 6.19: Balon modelinde çalıkuşundan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



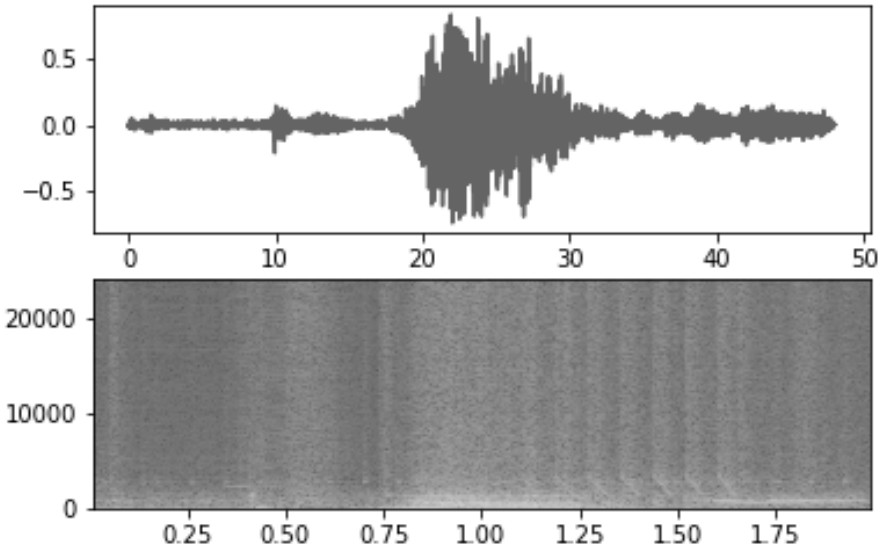
Şekil 6.20: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



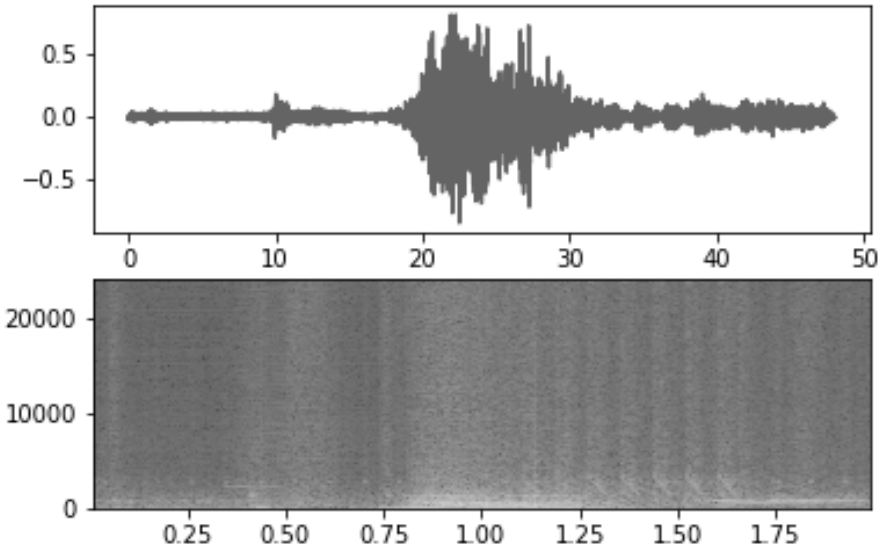
Şekil 6.21: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



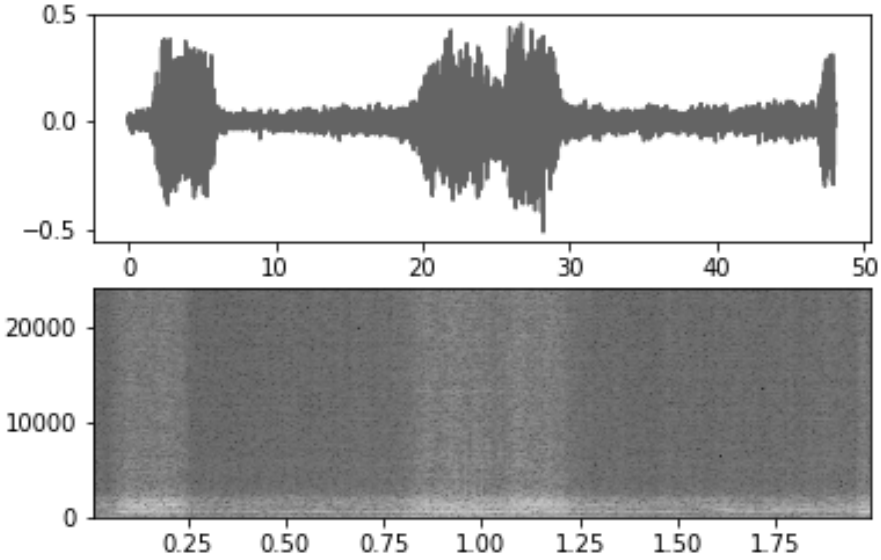
Şekil 6.22: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



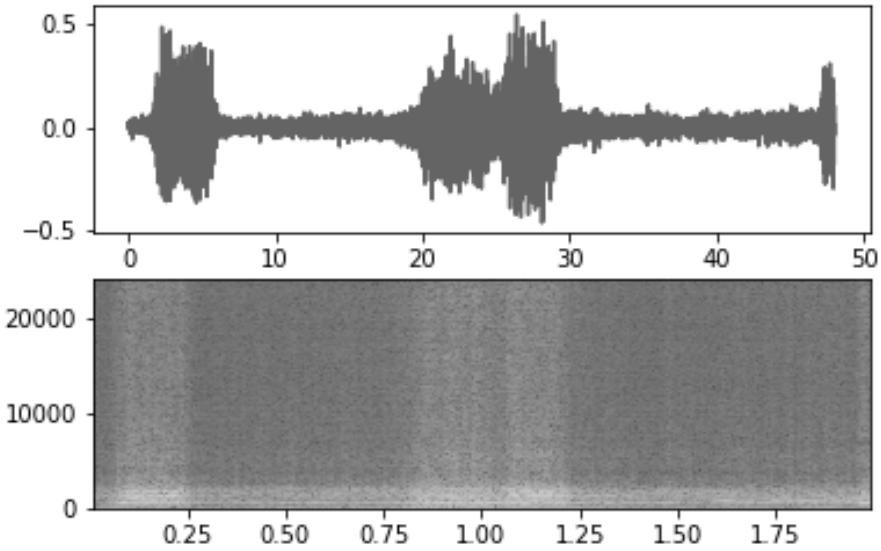
Şekil 6.23: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



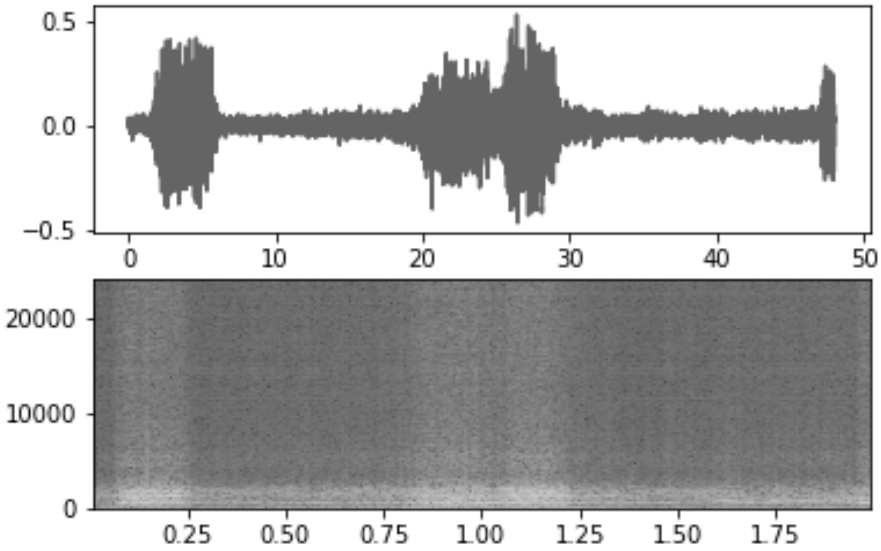
Şekil 6.24: Kova modelinde çalıkuşundan ayrılmış kaplan sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



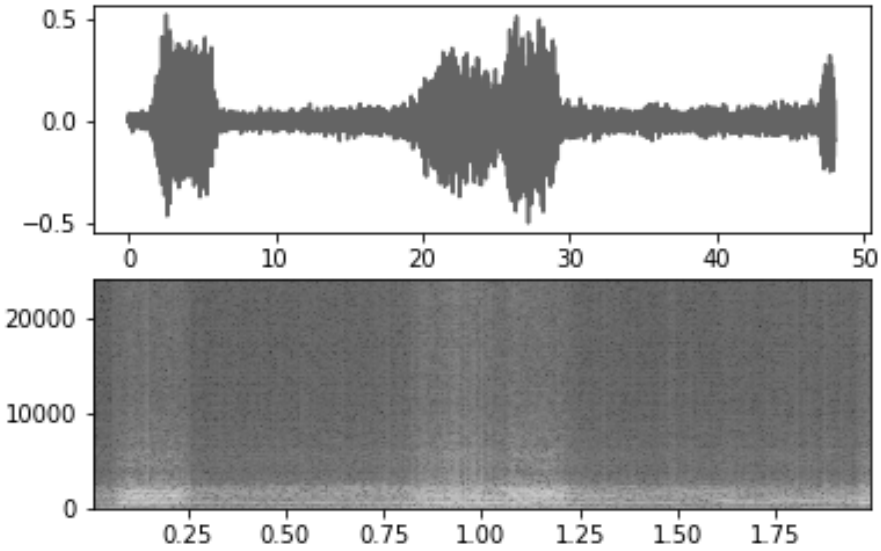
Şekil 6.25: Papyon modelinde kaplandan ayrılmış kangal sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



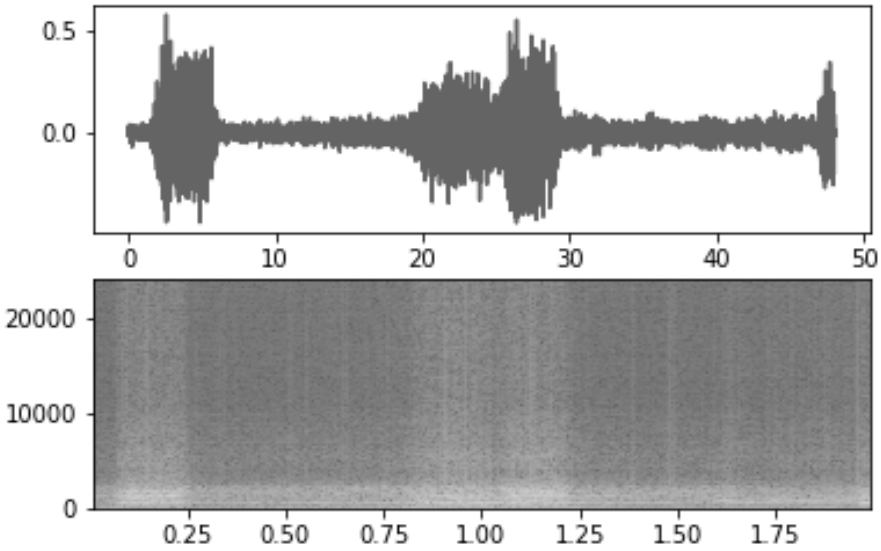
Şekil 6.26: Papyon modelinde kaplandan ayrılmış kangal sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



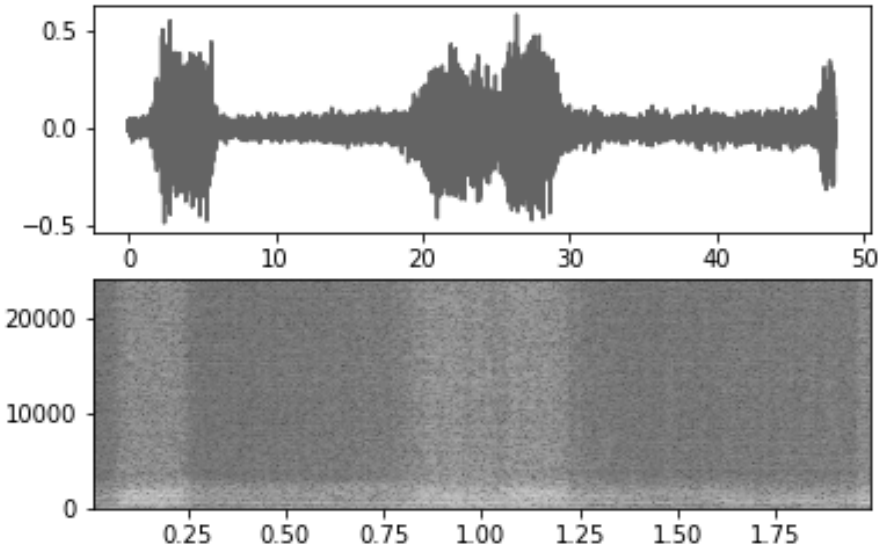
Şekil 6.27: Papyon modelinde kaplandan ayrılmış kangal sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



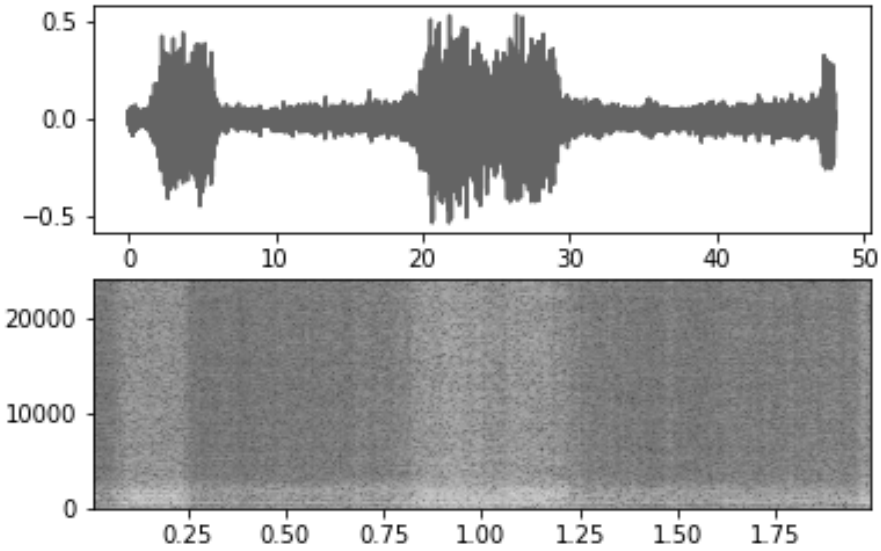
Şekil 6.28: Papyon modelinde kaplandan ayrılmış kangal sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



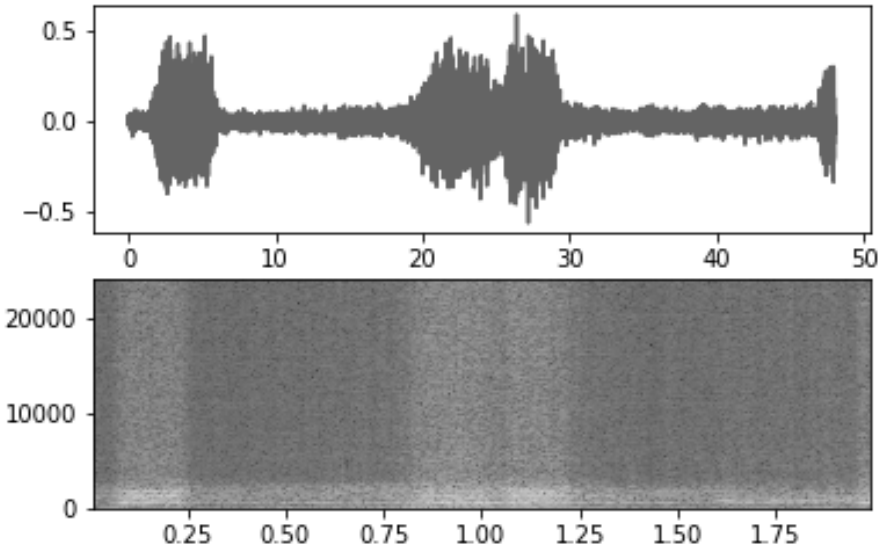
Şekil 6.29: Papyon modelinde kaplandan ayrılmış kangal sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



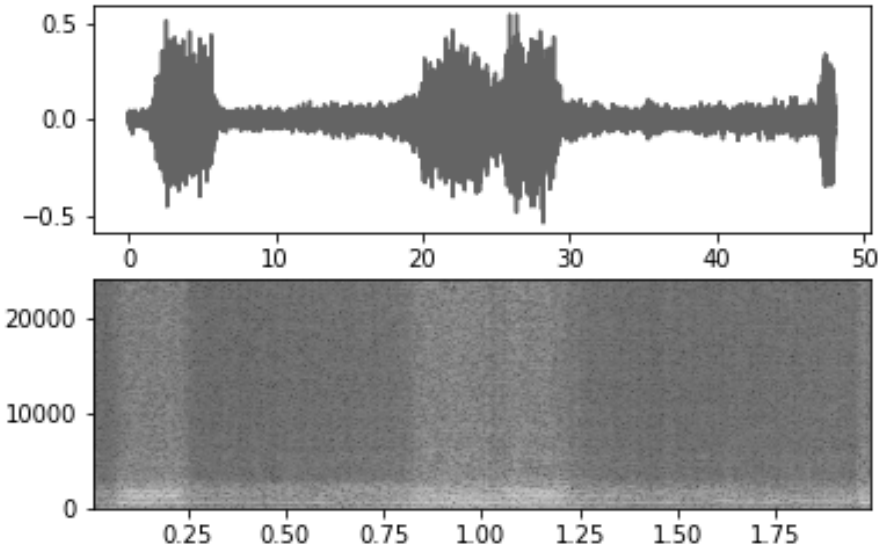
Şekil 6.30: Balon modelinde kaplandan ayrılmış kangal sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



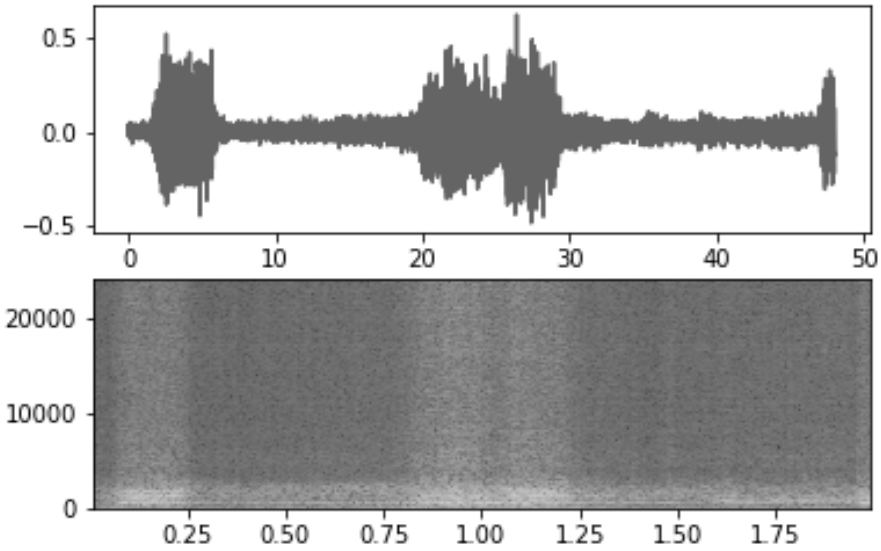
Şekil 6.31: Balon modelinde kaplandan ayrılmış kangal sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



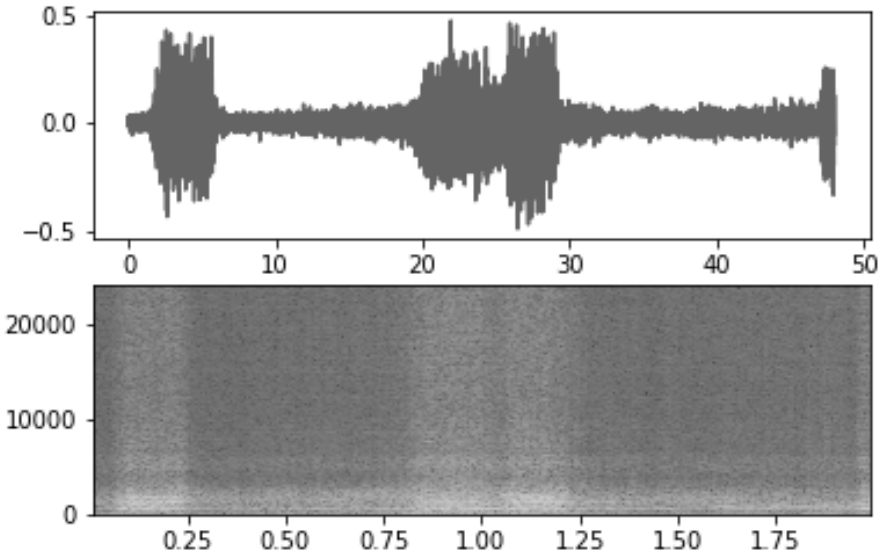
Şekil 6.32: Kova modelinde kaplandan ayrılmış kangal sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



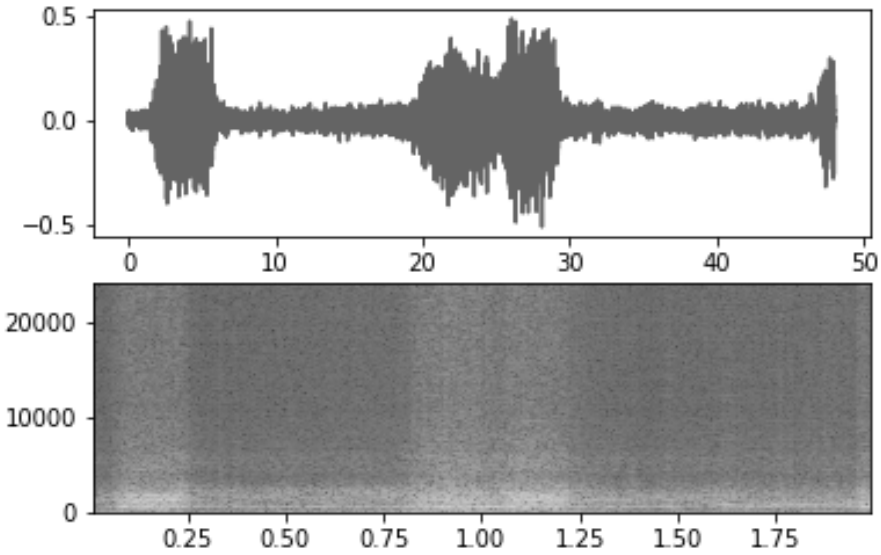
Şekil 6.33: Kova modelinde kaplandan ayrılmış kangal sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



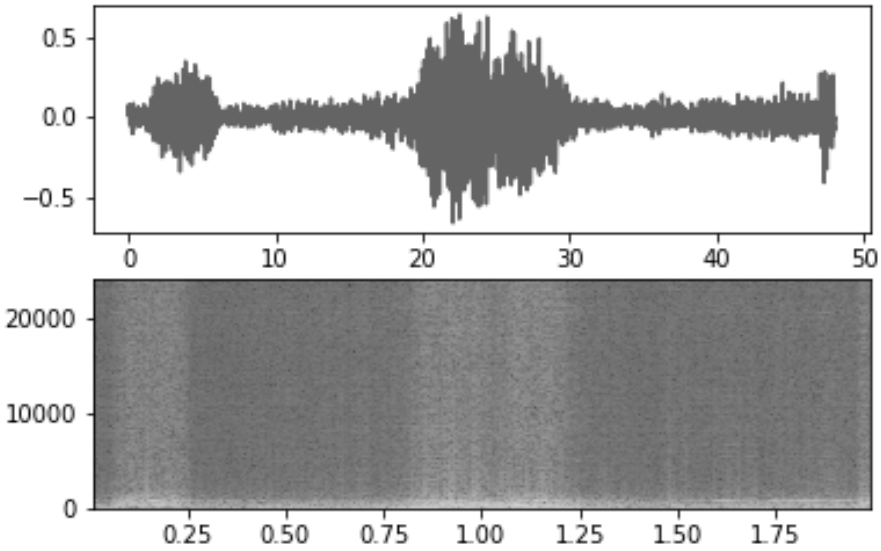
Şekil 6.34: Kova modelinde kaplandan ayrılmış kangal sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



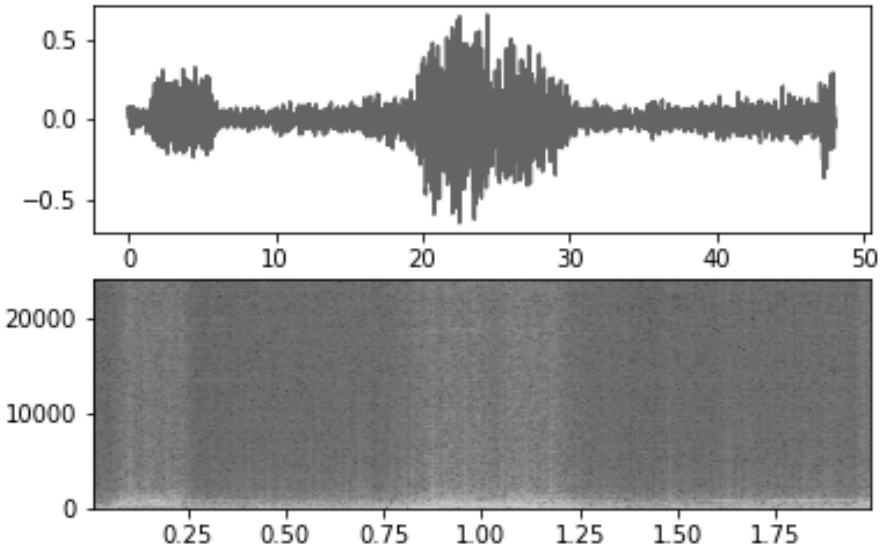
Şekil 6.35: Kova modelinde kaplandan ayrılmış kangal sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



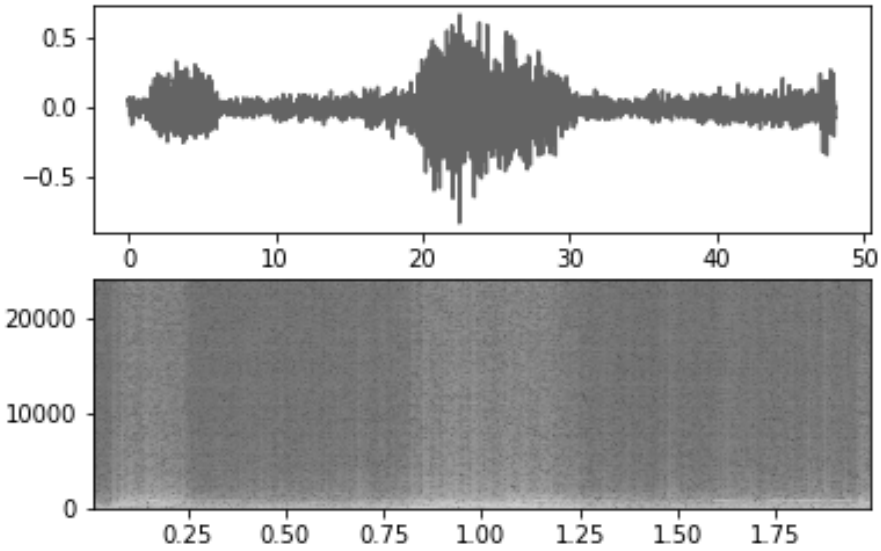
Şekil 6.36: Kova modelinde kaplandan ayrılmış kangal sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



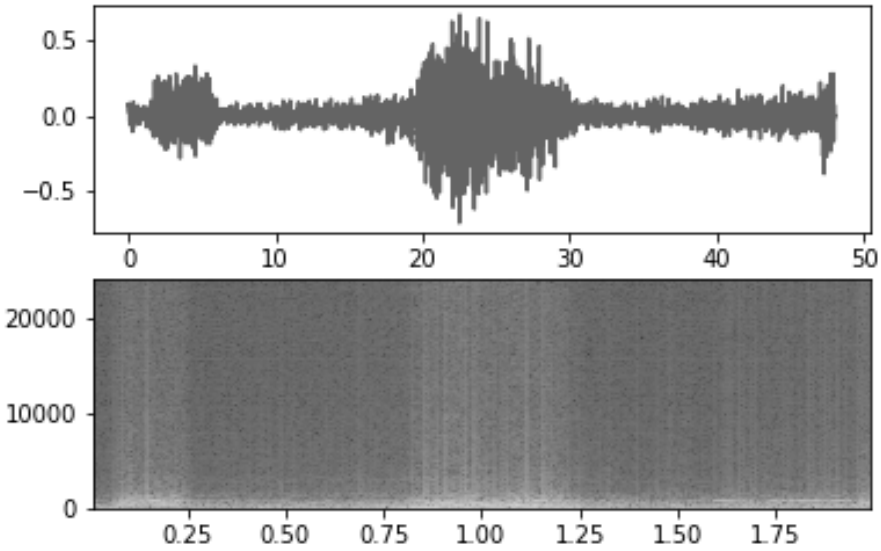
Şekil 6.37: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



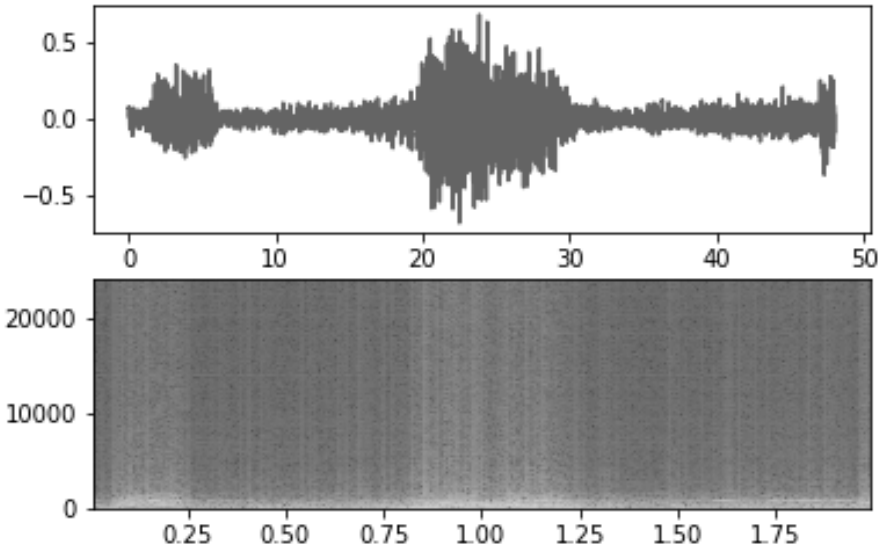
Şekil 6.38: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



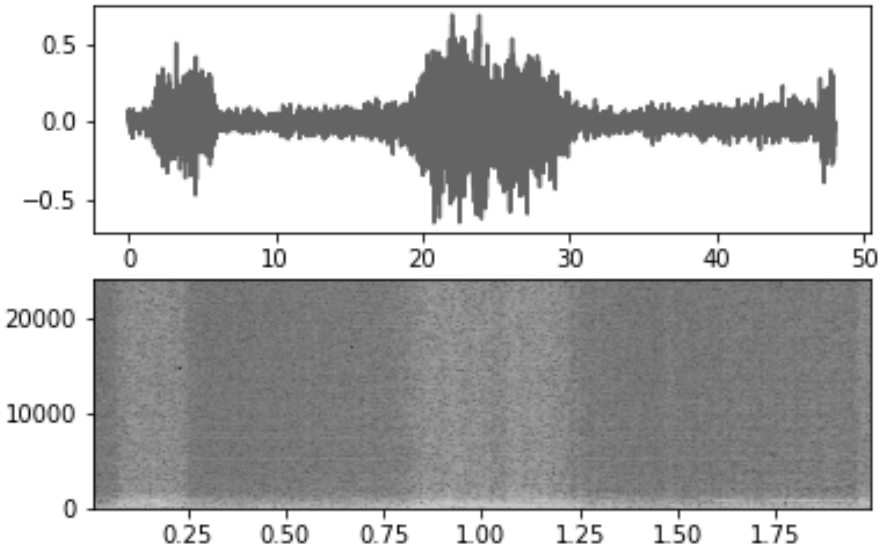
Şekil 6.39: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



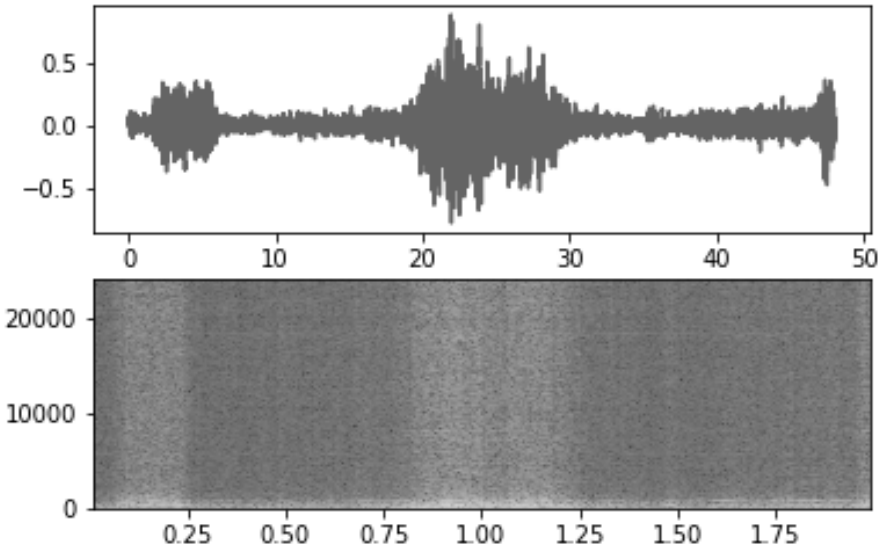
Şekil 6.40: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



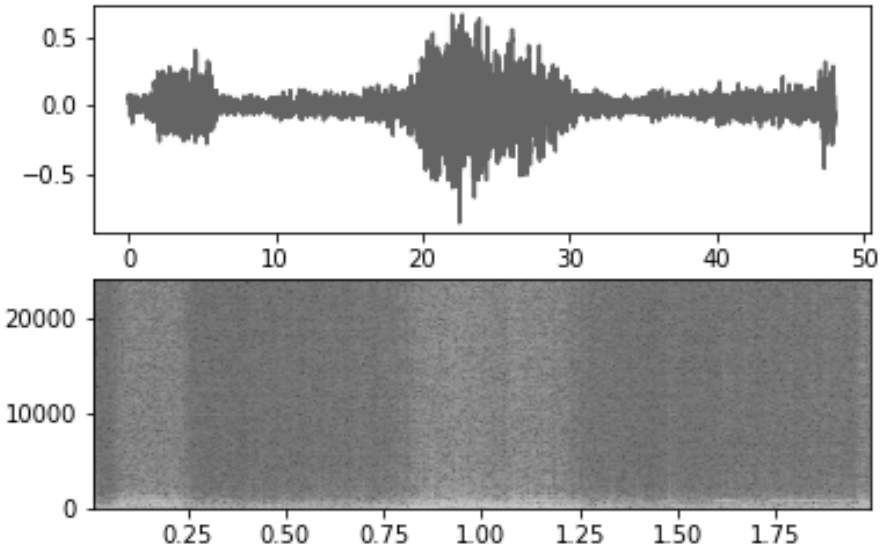
Şekil 6.41: Papyon modelinde kangaldan ayrılmış kaplan sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



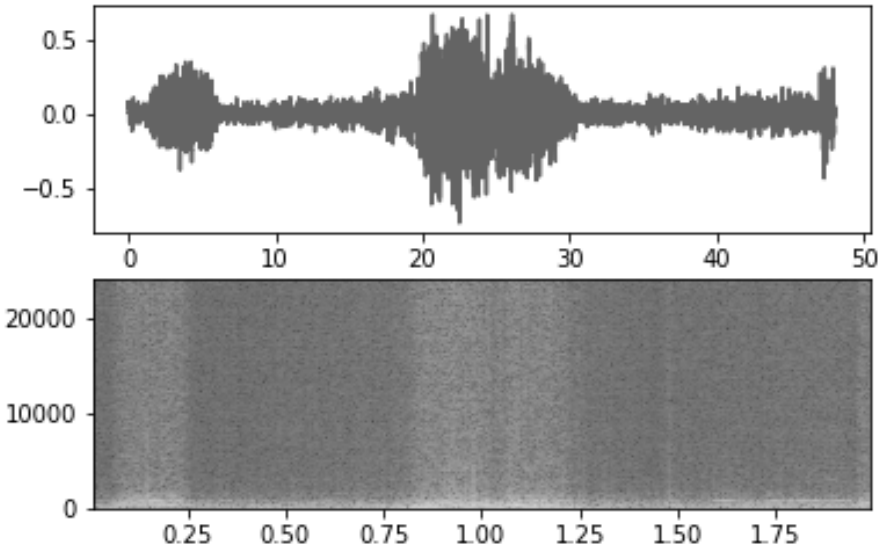
Şekil 6.42: Balon modelinde kangaldan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



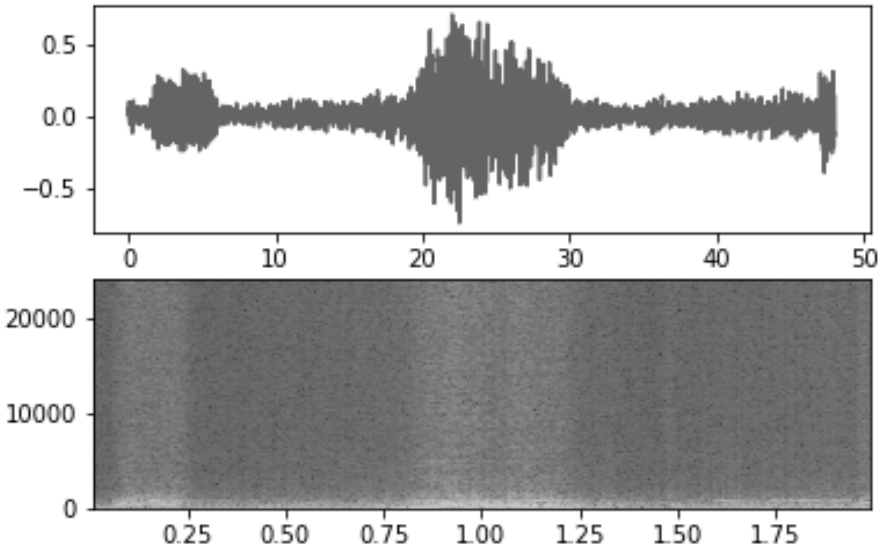
Şekil 6.43: Balon modelinde kangaldan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



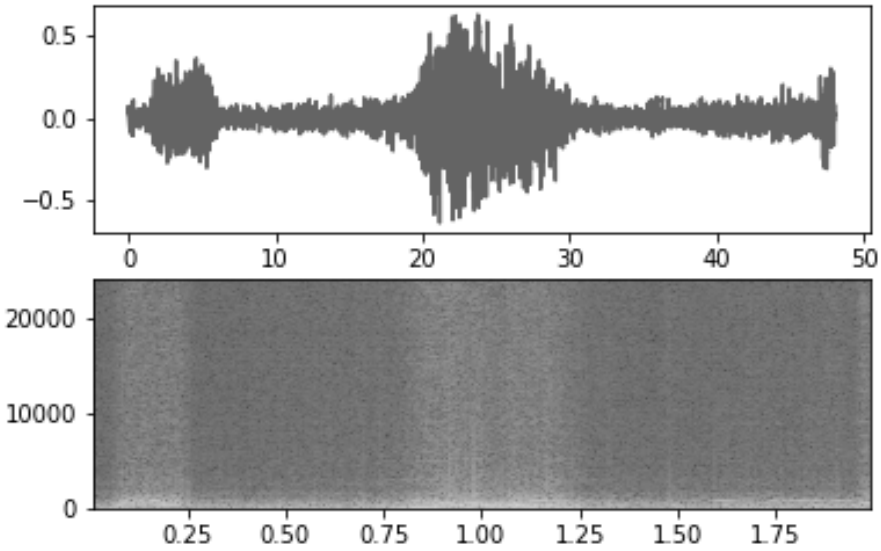
Şekil 6.44: Kova modelinde kangaldan ayrılmış kaplan sesinin, 5 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



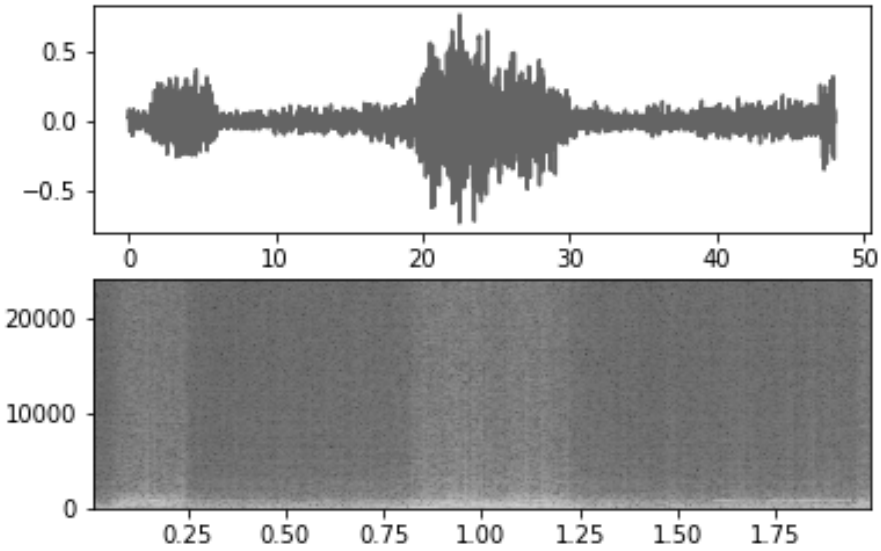
Şekil 6.45: Kova modelinde kangaldan ayrılmış kaplan sesinin, 10 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 6.46: Kova modelinde kangaldan ayrılmış kaplan sesinin, 15 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 6.47: Kova modelinde kangaldan ayrılmış kaplan sesinin, 30 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.



Şekil 6.48: Kova modelinde kangaldan ayrılmış kaplan sesinin, 60 çevrim eğitimden sonraki genlik-zaman grafiği üstte, spektrogram grafiği alttadır.

Kaynaklar

[1] Kenneth Hugdahl. What can be learned about brain function from dichotic listening. *Rev Esp Neuropsicol*, 2(3):62–84, 2000.

[2] Josh H McDermott. The cocktail party problem. *Current Biology*, 19(22):R1024–R1027, 2009.

[3] J. . Cardoso. Source separation using higher order moments. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 2109–2112 vol.4, May 1989.

[4] J. . Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44(12):3017–3030, Dec 1996.

[5] A. Belouchrani, K. Abed-Meraim, J. . Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, Feb 1997.

[6] A. Taleb and C. Jutten. Source separation in post-nonlinear mixtures. *IEEE Transactions on Signal Processing*, 47(10):2807–2820, Oct 1999.

[7] Tzyy-Ping JUNG, Scott Makeig, Colin Humphries, Te-Won Lee, Martin J. McKeown, Vicente Iragui, and Terrence J. Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178, 2000.

[8] Sam T Roweis. One microphone source separation. In *Advances in neural information processing systems*, pages 793–799, 2001.

[9] N. Linh-Trung, A. Aissa-El-Bey, K. Abel-Meraim, and A. Belounchrani. Underdetermined blind source seperation of non-disjoint nonstationary sources in the time-frequency domain. In *Proceedings of the Eighth International Symposium on Signal Processing and Its Applications, 2005.*, volume 1, pages 46–49, Aug 2005.

[10] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, July 2006.

[11] H. Tao, J. Zhang, and L. Yu. Adaptive blind source seperation using temporal predictability. In *2006 International Conference on Communications, Circuits and Systems*, volume 1, pages 280–283, June 2006.

[12] S. S. Kumar, S. H. Bharathi, and M. Archana. Non-negative matrix based optimization scheme for blind source separation in automatic speech recognition system. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–6, Oct 2016.

[13] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[14] Computing Machinery. Computing machinery and intelligence-am turing. *Mind*, 59(236):433, 1950.

[15] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[16] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[17] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[18] Stuart Dreyfus. Variational problems with inequality constraints. *Journal of Mathematical Analysis and Applications*, 4(2):297–308, 1962.

[19] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[20] Juyang Weng, Narendra Ahuja, and Thomas S Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 576–581. IEEE, 1992.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] L Deng, D Yu, and G Hinton. Deep learning for speech recognition and related applications. In *NIPS Workshop*, 2009.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[25] Bernard Marr. A Short History Of Deep Learning: Everyone Should Read. https://www.forbes.com/sites/bernardmarr/2016/03/22/a-short-history-of-deep-learning-everyone-should-read/76fb7e0e5561, 2016.

[26] Keith D. Foote. A Brief History of Deep Learning. https://www.dataversity.net/brief-history-deep-learning/, 2017.

[27] Keith D. Foote. A Brief History of Artificial Intelligence. https://www.dataversity.net/brief-history-artificial-intelligence/, 2016.

[28] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, USA, 1st edition, 1997.

[29] Shivam Bansal. How Autoencoders Work - Understanding the Math and Implementation. https://www.kaggle.com/shivamb/how-autoencoders-work-intro-and-usecases, 2018.

[30] Gabriel Castaneda, Paul Morris, and Taghi M Khoshgoftaar. Evaluation of maxout activations in deep learning across several big data domains. *Journal of Big Data*, 6(1):72, 2019.

[31] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[32] Python: Mean Squared Error. https://www.geeksforgeeks.org/python-mean-squared-error/, 2019.

[33] Peak Signal-to-Noise Ratio as an Image Quality Metric. https://www.ni.com/en-tr/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html, 2019.

[34] Welcome to Python.org. https://www.python.org, 2019.

[35] Welcome To Colaboratory. https://colab.research.google.com/notebooks/welcome.ipynb, 2019.

[36] Project Jupyter: About Us. https://jupyter.org/about, 2019.

[37] Jupyter And The Future Of The IPython: IPython. https://ipython.org, 2019.

[38] Colaboratory - Frequently Asked Questions. https://research.google.com/colaboratory/faq.html, 2019.

[39] Tensorflow: An end-to-end open source machine learning platform. https://www.tensorflow.org, 2019.

[40] Keras: The Python Deep Learning library. https://keras.io, 2019.

[41] NumPy: About NumPy. https://numpy.org, 2019.

[42] PyDrive 1.3.1. https://pypi.org/project/PyDrive/, 2019.

[43] Google Drive. https://www.google.com/drive/, 2019.

[44] WAVE Audio File Format. https://www.loc.gov/preservation/digital/formats/fdd/fdd000001.shtml, 2012.

[45] WAV File. https://tinyurl.com/tj5vwbn, 2019.

[46] Microsoft Corporation IBM Corporation. Multimedia Programming Interface and Data Specifications 1.0. https://tinyurl.com/rhk8ufy, 1991.

[47] Microsoft Corporation Laile L. Di Silvestro (Microsoft), Greg Baribault (Microsoft). Waveform Audio File Format MIME Sub-type Registration. https://tools.ietf.org/html/draft-ema-vpim-wav-00, 1999.

[48] YouTube. https://www.youtube.com, 2019.

[49] The Cornell Lab of Ornithology: Macaulay Library. https://www.macaulaylibrary.org, 2019.

[50] The Cornell Lab of Ornithology: eBird. https://ebird.org/home, 2019.

[51] Davinci Resolve 15. https://www.blackmagicdesign.com/tr/products/davinciresolve/, 2019.

[52] Linear Pulse Code Modulation (LPCM). https://www.techopedia.com/definition/10707/linear-pulse-code-modulation-lpcm, 2019.

[53] Linear Pulse Code Modulated Audio (LPCM). https://www.loc.gov/preservation/digital/formats/fdd/fdd000011.shtml, 2019.

[54] SoundFile 0.10.3.post1. https://pypi.org/project/SoundFile/, 2019.

[55] Fulvous Wren. https://macaulaylibrary.org/asset/91625, 2019.

[56] Tiger Sounds. https://www.youtube.com/watch?v=SkFx45lxlvA, 2019.

[57] Kangal Sesi. https://www.youtube.com/watch?v=UkmYZavAeXg, 2019.

[58] Peak Signal to Noise Ratio (PSNR) in Python for an Image. https://dsp.stackexchange.com/questions/38065/peak-signal-to-noise-ratio-psnr-in-python-for-an-image, 2019.

[59] Spectrogram Demo. https://dsp.stackexchange.com/questions/38065/peak-signal-to-noise-ratio-psnr-in-python-for-an-image, 2019.

[60] matplotlib.axes.Axes.specgram. https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.axes.Axes.specgram.html, 2019.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ÖZBAYGIN, Ahmet Sinan
Uyruğu	: TC
Doğum tarihi ve yeri	: 23/05/1992 - Konya
Medeni Hali	: Bekar
Tel	: +90 506 735 73 56
Fax	: -
E-mail	: sinan.ozbaygin@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek Lisans	: KTO Karatay Üniversitesi	-
Lisans	: KTO Karatay Üniversitesi	Temmuz-2015
Lise	: Konya Selçuklu Lisesi (Konevi Anadolu Lisesi)	Haziran-2010

İş Deneyimi

Yıl	Yer	Görev
2012-2013	Bürotime	Stajyer Mühendis
2011-2012	KOSKİ	Stajyer Mühendis

Yabancı Dil

İngilizce (İyi), Japonca (Başlangıç)

Yayınlar

Makale(ler)

- Derin Öğrenme ile Kaynak Ayrıştırma - Ahmet Sinan Özbaygın, Hüseyin Oktay Altun -

Bildiri(ler)

- Derin Öğrenme ile Kaynak Ayrıştırma - Ahmet Sinan Özbaygın, Hüseyin Oktay Altun -