KÜTAHYA SAĞLIK BİLİMLERİ ÜNİVERSİTESİ

Yapay Zeka Tabanlı Ses Üretimi

Beyzanur Dağdelen

Final Raporu

Danışman

Dr.Öğr.Üyesi Emre Güngör

Yapay Zeka & Veri Odaklı Sistem Tasarımı & Latex İle Rapor Hazırlama

14.06.2024

İçindekiler

Özet

Son yıllarda, derin öğrenme modelleri, doğal dil işleme ve görüntü işleme gibi çeşitli alanlarda önemli ilerlemeler kaydetmiştir. Bu modeller, büyük miktarda veriden öğrenme ve karmaşık görevleri gerçekleştirme yeteneğine sahiptir.

Ses üretimi, insan-bilgisayar etkileşimi ve sanal gerçeklik gibi birçok uygulamada önemli bir rol oynamaktadır. Geleneksel ses üretim yöntemleri, genellikle sentetik ve gerçekçi olmayan sesler üretmektedir.Özellikle, üretici adversarial ağlar (GAN'ler) ve varyasyonel otoenkoderler (VAE'ler) ses verilerinin işlenmesi ve üretilmesi konusunda güçlü araçlar olarak öne çıkmaktadır.

Bu çalışma, üretici ağlardan VAE kullanarak daha doğal ve gerçekçi sesler üretebilmeyi sunmaktadır.

Bu çalışmanın amacı, birkaç ses dataseti üzerinde eğitilmiş bir VAE modelinin, verilen girişlere göre gerçekçi ve kaliteli ses çıktıları üretebilme yeteneğini incelemektir.

Belirli bir ses datasetini kullanarak VAE modeli eğitilmiş ve modelin çıktıları çeşitli metriklerle değerlendirilmiştir.

Anahtar kelimeler

- Varyasyonel Otoenkoder
- Üretici Ağlar
- Ses Üretimi
- Derin Öğrenme
- Ses Dataseti
- Yapay Zeka

1 Giriş

Bu proje, yapay zeka ile ses üretimine odaklanan üç farklı çalışmayı incelemektedir. İlk olarak OpenAI Jukebox projesi, ardından Valerio Velardo'nun "The Sound of AI" YouTube kanalındaki proje ve son olarak Kaggle üzerindeki müzik üretim projesi ele alınacaktır.
Projenin ilk haftasında, yapay zeka ile ses üretimi ve sanal gerçeklik teknolojisi kullanılarak insanları huzurlu hissettiren ortamlar yaratılacaktır. Hedef, iyileştirici etkiye sahip farklı frekanslardaki sesleri yapay zeka ile üreterek, insanların olumsuz duygularını azaltmak ve motivasyon sağlayarak hayatlarını iyileştirmektir.
Bu hedeflere ulaşmak için, VQ-VAE (Vector Quantized-Variational Autoencoder) gibi ileri yapay zeka tekniklerinden faydalanılabilir. VQ-VAE, seslerin temsili ve yeniden oluşturulmasında kullanılabilir. Bu teknik, seslerin duygusal etkilerini analiz ederek, kullanıcıların duygusal durumlarını iyileştirmek için özelleştirilmiş sesler oluşturabilir. Ayrıca, bu

seslerin Unreal Engine gibi sanal gerçeklik ortamlarında kullanılmasıyla, kullanıcılar huzur ve motivasyon sağlayan görsel ve işitsel deneyimler yaşayabilirler. Böylece, VQ-VAE ve sanal gerçeklik teknolojileri bir araya getirilerek, insanların duygusal refahlarını artırmak için yenilikçi bir yaklaşım sunulabilir.

2 Literatür Araştırması

Bu bölüm, yapay zeka ile ses üretimine odaklanan üç çalışmayı incelemektedir: OpenAI Jukebox, Valerio Velardo'nun "The Sound of AI" YouTube kanalı ve Kaggle üzerindeki müzik üretim projeleri. Projenin amacı, yapay zeka ve sanal gerçeklik teknolojilerinin birleşimiyle iyileştirici etkileri olan sesler üretmek ve bu sesleri kullanıcıların duygusal refahını artırmak için kullanmaktır. VQ-VAE (Vector Quantized-Variational Autoencoder) tekniği, bu seslerin temsili ve yeniden oluşturulmasında kullanılacak ve özelleştirilmiş sesler oluşturulacaktır. Bu sesler, Unreal Engine gibi sanal gerçeklik ortamlarında kullanılarak kullanıcılar için huzur ve motivasyon sağlayan deneyimler sunacaktır.

OpenAI Jukebox

OpenAI Jukebox, insan yapımı şarkılara benzeyen yeni müzik parçaları üretebilen bir yapay zeka projesidir. Derin öğrenme algoritmaları kullanarak geniş bir müzik veri kümesi üzerinde eğitilmiş olan model, çeşitli müzik türlerinde ve tarzlarda orijinal parçalar oluşturabilir. Jukebox, sadece melodiler ve armoniler değil, aynı zamanda şarkı sözleri de üretebilir. Kullanıcılar, belirli bir sanatçı, tür veya ruh hali gibi kriterlere göre müzik oluşturmak için Jukebox'ı yönlendirebilirler. Jukebox'ın dikkat

çeken özellikleri arasında geniş müzik yelpazesi, şarkı sözleri üretme yeteneği, kullanıcı tarafından özelleştirilebilir olması ve yüksek kaliteli sonuçlar yer almaktadır. Bu proje, müziğin sınırlarını genişletirken yapay zeka alanında büyük bir ilerleme sunmaktadır[1].

Valerio Velardo'nun "The Sound of AI" YouTube Kanalı

Valerio Velardo'nun "The Sound of AI" isimli YouTube kanalındaki proje, sinir ağlarıyla ses oluşturma konusunu ele almaktadır. Bu projede, değişken otomatik kodlayıcılar kullanarak ses üretimi ve eğitim süreçleri tanıtılmaktadır. TensorFlow ve Keras kullanarak kodlama yapmanın yanı sıra, ses veri setlerinin ön işlenmesi ve kısa süreli Fourier dönüşümü gibi işlemler de detaylı olarak anlatılmaktadır. Velardo'nun projesi, varyasyonel oto-kodlayıcılarla ses üretimi, gerekli ön bilgiler ve kullanılacak teknoloji yığını hakkında kapsamlı bilgiler sunar. Proje, Python, TensorFlow, Keras ve Librosa gibi araçlar kullanılarak uygulamalı ve teorik bilgilerle desteklenmektedir[25].

Kaggle Üzerindeki Müzik Üretim Projesi

Kaggle üzerindeki proje, OpenAI Jukebox araştırma makalesinden ilham alınarak geliştirilmiştir [33]. Bu projede, belirli müzik kategorilerinden (jazz ve klasik müzik) müzik üretebilecek bir varyasyonel oto-kodlayıcı modeli geliştirilmiştir. Model, TensorFlow kullanılarak uygulanmış ve ses işlemleri için Librosa kütüphanesi kullanılmıştır. Proje, varyasyonel oto-kodlayıcıların çalışma prensiplerine, kısa süreli Fourier dönüşümüne ve ses veri setlerinin ön işlenmesine odaklanmaktadır. Model, müzik üretimi için optimize edilmiş olup, gelecekte daha fazla müzik kategorisi ile eğitilerek daha geniş bir yelpazede müzik üretebilecek şekilde geliştirilebilir.

Bu projede, yapay zekanın müzik üretiminde nasıl kullanılabileceği ve varyasyonel oto-kodlayıcıların potansiyeli üzerinde durulmuştur.
Bu üç çalışma, yapay zeka ile ses üretiminin farklı yönlerini ve uygulamalarını ele alarak, projenin temelini oluşturmaktadır. Yapay zeka ve sanal gerçeklik teknolojilerinin entegrasyonu ile iyileştirici etkileri olan sesler üretmek ve bu sesleri kullanıcıların duygusal refahını artırmak için kullanmak mümkündür. Bu çalışmalar, seslerin duygusal etkilerini analiz etmek ve özelleştirilmiş sesler oluşturmak için gerekli teknik ve teorik bilgileri sağlamaktadır.

2.1 Ses Datasetleri

- N-synth
- MAESTRO

2.1.1 N-synth

Benzersiz perde ve tınıya sahip 305.979 müzik notası içeren, tek seferlik enstrümantal notalardan oluşan bir veri kümesidir. Sesler, ticari örnek kütüphanelerindeki 1006 enstrümandan toplanmıştır ve kaynaklarına (akustik, elektronik veya sentetik), enstrüman ailesine ve sonik niteliklerine göre açıklanmıştır. Açıklamada kullanılan enstrüman aileleri bas, pirinç, flüt, gitar, klavye, tokmak, org, kamış, yaylı, synth lead ve vokaldir. Enstrümanlar için dört saniyelik monofonik 16kHz ses parçacıkları (notalar) oluşturulmuştur [6].

2.1.2 MAESTRO

MAESTRO (MIDI and Audio Edited for Synchronous Tracks and Organization), nota etiketleri ve ses dalga formları arasında ince bir hizalama (3 ms) ile yakalanan 200 saatin üzerinde virtüöz piyano performansından oluşan bir veri kümesidir [7]. MIDI, dijital enstrümanların birbirleriyle 'mesajlar' aracılığıyla iletişim kurmasını sağlayan bir protokoldür. Bu mesajlar, çalma için kullanılacak enstrümanın türü, çalınacak notalar, bir notanın ne zaman başlayacağı, bir notanın ne zaman biteceği vb. hakkında bilgi depolar.

2.2 Kaglle Kodu İle Kullanılan Datasetler

- FSDD-Free Spoken Digit Dataset
- Gunshot Audio Dataset

2.2.1 FSDD - Free Spoken Digit Dataset

Free Spoken Digit Dataset (FSDD), 8kHz'de wav dosyalarında konuşulan rakamların kayıtlarından oluşan basit bir ses/konuşma veri kümesidir. Kayıtlar, başlangıçlarında ve sonlarında neredeyse minimum sessizliğe sahip olacak şekilde kırpılmıştır. Bu veri seti 6 konuşmacıdan, 3.000 kayıttan (konuşmacı başına her rakamdan 50 adet) ve İngilizce telaffuzlardan veri içermektedir [31].

8kHz'de wav dosyalarında konuşulan rakamların kayıtlarından oluşan basit

bir ses/konuşma veri kümesidir. Kayıtlar, başlangıç ve bitişlerinde neredeyse minimum sessizliğe sahip olacak şekilde kırpılmıştır.
FSDD açık bir veri setidir.
Mevcut durum

- 6 hoparlör
- 3.000 kayıt (konuşmacı başına her rakamdan 50 adet)
- İngilizce telaffuzlar

Organizasyon: Dosyalar aşağıdaki formatta adlandırılır:
digitLabel_speakerName_index.wav
Örnek: 7_jackson_32.wav

2.2.2 Gunshot Audio Dataset

Veri kaynağı olarak YouTube kullanılmaktadır. Silah seslerini toplamak için değişken ortam modeli kullanılmıştır. Silah modellerinin sesleri YouTube üzerinden herkese açık olan videolar kullanılarak toplanmıştır. Farklı tarihlerde toplanan bu dosyalar indirilerek her silah tipi için 2 saniyelik parçalar üretilmiştir. Bu şekilde 8 silah modeli ile toplam 851 dosya tipi elde edilmiştir. Ses dosyalarının örnekleme hızı 44100 Hz'dir. Öncelikle her bir ses dosyası wav dosya formatına dönüştürülmüştür. Daha sonra sesler dikkatlice dinlenmiş ve içerisinde farklı sesler veya gürültüler olmadığından emin olunmuştur. Ses dosyalarının parçalanması sırasında aynı sesin tekrar tekrar devam etmediği dikkatlice kontrol edilmiştir. Tüm bu işlemler için WavePad Audio Editor programı kullanılmıştır. Tablo 1'de oluşturulan seslerin sayısına ilişkin detaylar listelenmiştir [32].

Toplanan silah ses veri setinin detayları

Tablo 1: Silah Ses Verisi

Kimlik	**Model**	**Gözlem Sayısı**
1	AK-47	72
2	IMI Desert Eagle (Çöl Kartalı)	100
3	AK-12	98
4	M16	200
5	M249	99
6	MG-42	100
7	MP5	100
8	Zastava M92	82

3 Yöntem

Bu projede, derin öğrenme alanında öne çıkan ve çeşitli uygulamalarda başarıyla kullanılan VAE (Variational Autoencoder) ve türevleri araştırılmıştır. Araştırmalar sonucu VAE kullanılmıştır. Üretici ağlar olarak bilinen bu modeller, veri üretimi ve temsili öğrenme konularında güçlü araçlar sunmaktadır. Projede ayrıca, model performansını artırmak ve en iyi sonuçları elde etmek amacıyla çeşitli optimizasyon algoritmaları (Adam) kullanılmıştır. Kullanılan loss fonksiyonları (KL Diverjansı) ise, modelin doğruluğunu ve genelleme yeteneğini maksimize etmek için özenle seçilmiş ve optimize edilmiştir. Bu yöntemler, projenin hedeflerine ulaşmasında ve başarılı sonuçlar elde edilmesinde kritik rol oynamıştır.

3.1 Variational Autoencoder (VAE): Yaratıcı ve Esnek Veri Üretimi

Variational Autoencoder (VAE), veri temsilini öğrenmek ve yaratıcı bir şekilde yeni veri noktaları üretmek için kullanılan önemli bir modeldir. VAE, derin öğrenme modelleri sınıfında yer alan bir tür yapay sinir ağıdır. Amacı, veri setindeki örüntüleri öğrenerek, veriyi daha az boyutta bir temsile sıkıştırmak ve bu temsili kullanarak yeni veri noktaları oluşturmaktır. Genellikle denetimsiz öğrenme yöntemi olarak kullanılan VAE, veri setinin içsel yapısını anlama yeteneğine sahiptir [2].

3.1.0.1 VAE'nin İşlevi ve Autoencoder (AE) ile Farkı

VAE, geleneksel bir autoencoder (AE) gibi çalışır, ancak belirgin bir farklılık bulunur. AE, gizli bir temsil (latent representation) oluşturmak için veriyi sıkıştırır ve ardından bu temsil aracılığıyla veriyi yeniden oluşturur. VAE ise sadece veriyi sıkıştırmakla kalmaz, aynı zamanda verinin olası dağılımını modelleyerek, veri noktalarının olasılık dağılımını öğrenir. Bu da VAE'nin veri üretiminde daha esnek ve çeşitlilik yaratabilme yeteneğine sahip olmasını sağlar.

3.1.1 VAE'nin Çalışma Prensibi

VAE'nin temel prensibi, veriye gizli bir yapısal temsil oluşturmak ve bu temsil aracılığıyla yeni veri noktaları üretmektir. Bu süreç iki ana bileşen üzerine kuruludur: encoder ve decoder. Veriyi latent uzayın parametreleri

olan bir dağılımın parametreleri haline getiren encoder ağıdır. Bu dağılım genellikle Gaussian (normal) dağılımı olarak belirtilir. Encoder, veriyi

sıkıştırarak, her bir veri noktasını latent uzayın içindeki bir noktaya eşler. Latent uzaydaki noktaları orijinal veriye çeviren ise decoder ağıdır. Encoder

tarafından elde edilen latent uzay temsilini alır ve bu temsili orijinal veriye dönüştürür.

3.1.2 GAN ile Bağlantısı

GAN, gerçekçi veri üretmek için kullanılırken, VAE daha yapılandırılmış ve kontrollü bir şekilde veri üretir. Bazı çalışmalar, VAE'nin öğrendiği temsili GAN yapısında kullanarak daha kaliteli ve çeşitli veri üretimi sağladığını göstermiştir.

3.1.3 Kullanım Alanları

Kullanım alanları geniş olan VAE'ye şunlar örnek verilebilir:

- Görüntü ve Video İşleme: Görüntü tamamlama, yeniden oluşturma ve sentezleme işlemlerinde kullanılabilir.
- Doğal Dil İşleme: Dil modelleme, metin üretimi ve dilin öğrenilmesi gibi alanlarda kullanılabilir.
- Yaratıcı İçerik Üretimi: Sanat, müzik ve edebiyat gibi yaratıcı içeriklerin oluşturulmasında kullanılabilir.

Variational Autoencoder, derin öğrenme ve generatif modelleme alanındaki önemli bir gelişmedir. Verinin daha esnek bir temsilini öğrenerek hem veri üretiminde kullanılabilirliği hem de verinin içsel yapısını anlama yeteneği ile birçok alanda geniş bir kullanım potansiyeline sahiptir.

3.1.4 VAE & VQ-VAE & VAE-GAN

Yapay zeka tabanlı model türleri olan bu üç model, özellikle ses ve görüntü üretmede kullanılırlar [3].

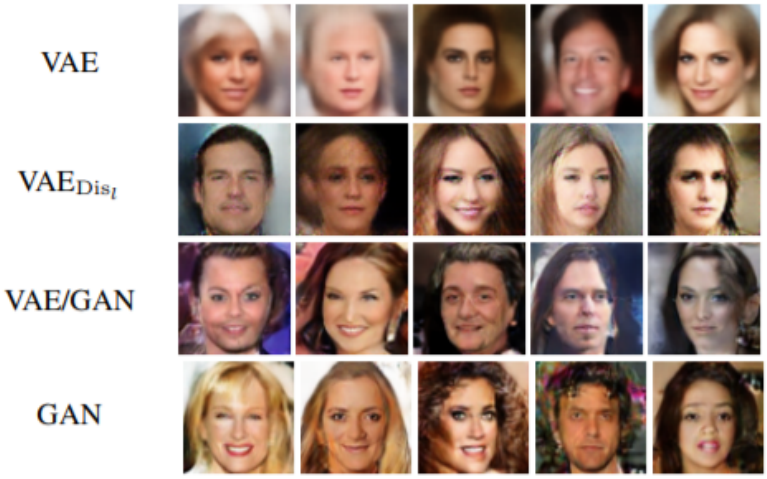


Figure 3. Samples from different generative models.

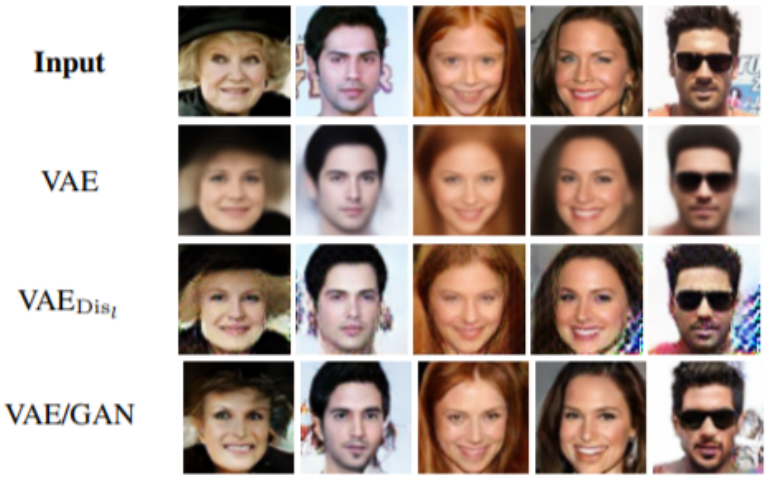
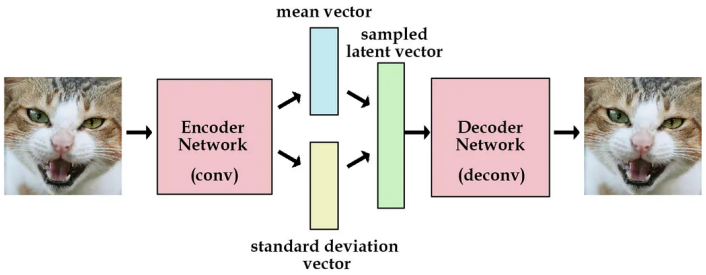


Figure 4. Reconstructions from different autoencoders.

Şekil 1: query&output

[3]

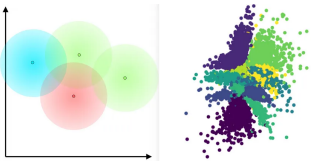
3.1.4.1 VAE Vanilya VAE en iyi bilinen üretken modellerden biridir (GAN dışında). Bir aşağı örnek, genellikle z olarak adlandırılan bir gizli vektör ve bir yukarı örnekten oluşan üç bölümü vardır. Aşağı örnek2 genellikle konvolüsyonlardan oluşur ve yukarı örnek ya transpoze edilmiş bir konvolüsyondur ya da normal bir yukarı örnek ve yukarı örnekten tutulan uzamsal boyutlarla konvolüsyon içeren bir bloktur. Bununla birlikte, konvolüsyonlu yukarı örnekleme genellikle normal transpoze konvolüsyonlardan daha düzgün sonuçlar üretir.



Şekil 2: VAE

[3]

3.1.4.2 VQ-VAE VQ-VAE, vanilya VAE'nin ayırt edici özelliği olan gizli uzayın yeniden yaratılmasıdır. Bu, gizli uzayı ortadan kaldırmak anlamına gelmemektedir, sadece sürekli yerine ayrık olmasına izin verir, dolayısıyla adın Vektör Niceleme VAE'dir. Gizli uzaydaki noktalar VQ-VAE'de ayrıktır ancak VAE'de süreklidir [4].
En yakın embeddingi ya da gizli uzayda örnekleme yapılacak en yakın noktayı hesaplayan bir fonksiyon ekleyerek bulanık çıktılar sorununu çözülmesi hedeflenmektedir. Çünkü VQ-VAE eğitildiği bir noktadan örnekleme alınırsa daha keskin olacaktır. VQ-VAE 2 görüntü kalitesi ve çoklu nesil çeşitliliği ölçütlerinde BIG-GAN'ı geride bırakmıştır. Bununla



Şekil 3: VQ-VAE

[4]

birlikte, aynı sonucu elde etmenin bir başka yolu da GAN'ın discriminatorunu kullanmaktır.

3.1.4.3 VAE-GAN GAN'lar bazen insanları bile kandırabilecek hiper-gerçekçi görüntüler üretebilmektedir fakat amaçlanan özelliklere sahip görüntüler üretmek için VAE'nin gizli uzay gibi bazı kısımlarına hala ihtiyaç duyulabilmektedir. Bunu yapmak için, GAN'ın ayırt edicisini VAE'nin dönüştürülmüş konvolüsyonlarından sonra eklemelidir. Görüntülerin daha keskin ve daha gerçekçi olmasına yardımcı olmak için VAE'nin loss fonksiyonuna bir ayırıcı kayıp bileşeni eklenmelidir [5].

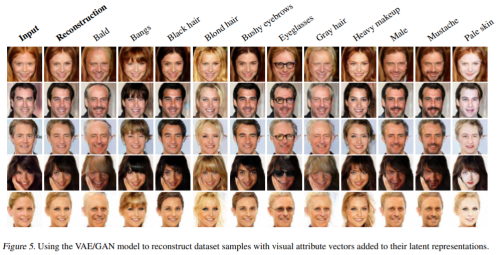


Figure 5. Using the VAE/GAN model to reconstruct dataset samples with visual attribute vectors added to their latent representations.

Şekil 4: VAE-GAN

[3]

3.2 Entropi

Entropi(düzensizlik), bir olayın veya iletinin içerdiği bilgi miktarıdır [11]. Bilgiyi ise bilgisayarlarda bitler halinde depolanır. Bilginin depolama işlemi gerçekleştirilirken minimum bit sayısı olacak şekilde saklanması amaçlanır. Olayların gerçekleşme olasılığı ile içerdiği bilgi miktarı karşılaştırılırsa;

- Düşük olasılıklı gerçekleşen olaylar büyük miktarda bilgi içermektedir.
- Yüksek olasılıkla gerçekleşen olaylar az sayıda bilgi içermektedir.

Bir olayın içerdiği bilgi miktarını formülize adildiğinde aşağıdaki(**??**) formül elde edilir.

$$Bilgi(x) = -log_2(p(x))[11] \tag{1}$$

Hilesiz bir madeni para için para yazı-tura oyunu için havaya atıldğında aşağıdaki tablo ortaya çıkmaktadır(2).

Tablo 2: Hilesiz Para Örneği [11]

Çıktılar	Yazı	Tura
Olasılık	0.5	0.5
Bit Gösterimi	0	1
Bit Sayısı	1	
Bilgi: $H(p) = -log_2(p(x))$	1	1

Bu olayların gerçekleşme olasılığı eşit olduğundan olaya dahi taşıdıkları bilgi miktarları birbirine eşit ve 2'dir.
Madeni para hileli olduğunda yazı ve tura gelme olasılıkları eşit değildir(3).

Tablo 3: Hileli Para Örneği [11]

Çıktılar	Yazı	Tura
Olasılık	0.25	0.75
Bit Gösterimi	0	1
Bit Sayısı	1	
Bilgi: $H(p) = -log_2(p(x))$	2	0.415

Hilesiz 2 adet bağımsız madeni paranın değerleri aşağıdaki gibidir(4);

Tablo 4: Hileli Para Örneği [11]

Çıktılar	Yazı-Yazı	Yazı-Tura	Tura-Yazı	Tura-Tura
Olasılık	0.25	0.25	0.25	0.25
Bit Gösterimi	00	01	10	11
Bit Sayısı	2			
Bilgi: $H(p) = -log_2(p(x))$	2	2	2	2

Olasılıkları farklı olan 4 olayı incelendiğinde(5);

Tablo 5: Hileli Para Örneği [11]

Çıktılar	X_1	X_2	X_3	X_4
Olasılık	0.5	0.25	0.125	0.125
Bit Gösterimi	1	01	000	001
Bit Sayısı	1	2	3	3
Bilgi: $H(p) = -log_2(p(x))$	1	2	3	3

Burada bit gösterimlerinin ve sayılarının farklı olduğu gözükmektedir.

Burada olaylar 3'e ayrılmış gibi görülebilmektedir. Yani p: olayın gerçekleşme olasılığı gösterilirse(3);

$$[!hb]P(X_1) = P(x_2) + P(x_3) + P(x_4) \tag{2}$$

$$[!ht]P(x_2) = P(x_3) + P(x_4) \tag{3}$$

şeklindedir.
İçerdikleri bilgi miktarlarına bakıldığında gerçekleşme olasılığı düşük olan olayların daha yüksek bilgi içerdikleri gözükmektedir.
İçerdiği bilgi hesaplamasını rastgele bir X değişkeni için hesaplarsak tüm gerçekleşen olaylar üzerinden beklenen bilgiye(expected information) bakılmaktadır. Entropi formüllerinden birisi olan Shannon Entropi aşağıdaki şekildedir(**??**).

$$H(X) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{4}$$

Sırayla bir adil ve bir adil olmayan madeni paraların yazı tura olaylarının entropileri ise(**??**);

Olaydaki belirsizlik, eşit olasılıklı yazı turaya göre daha düşüktür. Yani

Tablo 6: Bir Adil Madeni Para Örneği [11]

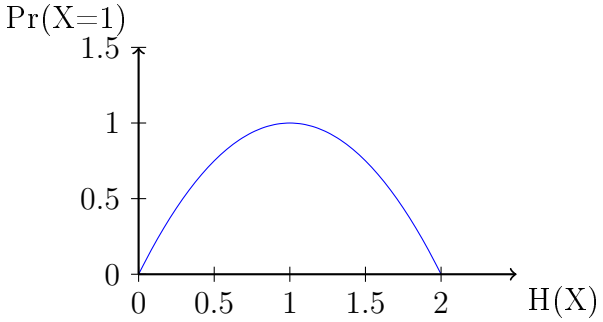
Çıktılar	Yazı	Tura
Olasılık	0.5	0.5
Bit Gösterimi	0	1
Bit Sayısı	1	
Bilgi: $H(p) = -log_2(p(x))$	1	1
Entropi: $: p * -log_2(p(x))$	0.5	0.5
Beklenen Entropi	1	

Tablo 7: Bir Adil Olmayan Para Örneği [11]

Çıktılar	Yazı	Tura
Olasılık	0.25	0.75
Bit Gösterimi	0	1
Bit Sayısı	1	
Bilgi: $H(p) = -log_2(p(x))$	2	0.415
Entropi: $: p * -log_2(p(x))$	0.5	0.307
Beklenen Entropi	0.807	

entropisi daha düşüktür.

P(X) dağılımı Bernoulli dağılımı olarak 2 ayrı x ve y olaylarının gerçekleştiği tanımlanılırsa Bernoulli dağılımı altında $p(x) = 1^{-}p(y)$ denilebilir. x olayının gerçekleşme olasılığı p(x) e göre entropi fonksiyonunun grafiği ise aşağıdaki şekildedir(3.2);



İki olayın gerçekleşme olasılığı eşit olduğunda (yani p(x)=0=1-p(y) dolayısıyla p(y)=0.5) entropinin maksimum değerini aldığı gözükmektedir. İki olayın gerçekleşme olasılıkları birbirine eşit olduğundan hangi sonucun geleceği diğer durumlara göre daha belirsizdir ve bu yüzden entropi maksimum değeri alır.

3.3 Çapraz Düzensizlik (Cross-Entropy) ve KL Iraksaklığı (KL Divergence)

Cross-Entropy, gerçek olasılık dağılımı P iken bulunan Q olasılık dağılımı için beklenen entropi değeridir. Cross-Entropy formülü (**??**);

$$H(p,q) = -\sum_{x} p(x) log_q(x) \tag{5}$$

Cross-Entropy'i Kullback-Leibler(KL) ıraksaklığını kullanarak da aşağıdaki şekilde gösterilebilmektedir;

$$H(p,q) = H(p) + D_w(p||q) \tag{6}$$

H(p), doğru olasılık dağılımı P'nin entropisidir. P doğru olasılık dağılımını bulmak için bir model kurulduğunda P'ye yaklaşık olan Q olasılık dağılımını vermektedir. Bu 2 olasılık dağılımı arasındaki farklılığa bakmak için tanımlanan ölçülerden biri Cross-Entropy fonksiyonunda gördüğünüz $D_kl(p||q)$ ölçüsü Kullback-Leibler(KL) ıraksaklığıdır.

3.4 Kullback-Leibler Iraksaklığı (KL-Divergence)

Makine öğrenmesi alanında P olasılık dağılımı yerine Q olasılık dağılımı kullanılırken elde edilen bilgi kazanım anlamınındayken, Bayes çıkarımında ise önceki dağılım Q'dan sonraki dağılım P'ye geçerken ki elde edilen bilgi kazanımıdır.Yani P olasılık dağılımını tahmin ederken Q olasılık dağılımı kullanıldığında kaybedilen bilgi miktarıdır. Formülünün olabilirlik oranından elde edilmesi ise adım adım aşağıdaki şekildedir(**??**).

$$LR = \frac{p(x)}{q(x)} \tag{7}$$

x değeri bilinmeyen bir olasılık dağılımından ise yukarıda görülen (**??**) olabilirlik oranı x örneklemi için P dağılımından gelmenin Q dağılımından gelmeye nazaran ne kadar olası olduğunu göstermektedir. Birden fazla bağımsız örneklemler için olabilirlik fonksiyonu hesaplanılırken her örneklem için bulunan olabilirlik oranının çarpılması gerekmektedir.

$$LR = \prod_{i=0}^{n} \frac{p(x_i)}{q(x_i)} \tag{8}$$

$$LR = \sum_{i=0}^{n} log(\frac{p(x_i)}{q(x_i))} \tag{9}$$

Örneklemde Q dağılımı üzerinde P'ye dair ortalama ne kadar bilgi verdiğini hesaplamak için olabilirlik oranının beklenen değeri**??**;

$$D_kl(P||Q) = \sum_{i=0}^{n} p(x_i)log(\frac{p(x_i)}{q(x_i))} \tag{10}$$

KL ıraksaklığı formülünü yukarıdaki şekliyle elde etmiş olunmaktadır. KL ıraksaklığının 0 olması P ve Q dağılımlarının aynı olduğunu ifade etmektedir. Yukarıdaki formül P ve Q dağılımları ayrık olasılık dağılımları için kullanılabilir(Örneğin Bernouilli, Poisson, Binom dağılımları).

$$D_kl(P||Q) = \int_{-\infty}^{\infty} p(x_i)log(\frac{p(x_i)}{q(x_i))}\, dx \tag{11}$$

Olasılık dağılımları sürekliyse formül yukarıdaki gibi integral şeklinde tanımlanır.

3.5 Optimizer

Optimizasyon algoritmaları, modelin kayıp fonksiyonunu minimize ederek modelin daha iyi performans göstermesini sağlar. Bu algoritmalar, ağırlıkları güncellemek için gradyan iniş (gradient descent) ve türev tabanlı teknikler kullanır. Optimizatörler, sinir ağı eğitiminde önemli bir rol oynar. Ağırlık ve önyargılar gibi parametreleri ayarlayarak ağın bir fonksiyonu en iyi şekilde temsil etmesini sağlarlar [13].

1. Gradyan İniş (Gradient Descent): Gradyan İniş (Gradient Descent), belki de en temel optimizasyon algoritmasıdır ve derin öğrenme modellerinin eğitiminde sıkça kullanılır. Amacı, kayıp fonksiyonunun ağırlıklar üzerinden türevini alarak, negatif gradyana doğru ağırlıkları güncellemektir [14].

2. Stochastic Gradient Descent: Stokastik gradyan azalma, gradyan azalma gibidir ancak gradyan azalmada bütün veri seti kullanılırken, stokastik gradyan azalmada tek bir örneklem kullanılır. Stokastik gradyan azalma yönteminde tek seçilen veri ile işlem yapıldığı için global mimimum noktasına kararlı bir şekilde ilerleme gözlenmez. Diğer taraftan, rastgele parametre katsayıları ile bir örnek üzerinde işlem yaptığından local minimaya düşme ihtimali daha düşüktür [14].

   Stokastik gradyan azalma, gradyan azalmaya nazaran daha gürültülü bir grafik çizer. Minima noktasına ulaşması için daha fazla iterasyon sayısına ihtiyaç duymaktadır ancak tek bir örnek üzerinde çalıştığı için, hesaplama maliyeti gradyan azalmadan daha verimlidir.

3. Uyarlanabilir Gradyan Algoritması (Adagrad): Adagrad, parametre güncellemelerini her bir parametre için ayrı ayrı ölçeklendirerek gradyan iniş optimizatörünün bir modifikasyonudur . Minibatch Gradyan azalmada başarıyı etkileyen faktörlerden biride kullanılan adımın boyutuydu. Bu adım boyutunu doğru belirleyememek, bizi minimuma yakınsamama problemi ile karşı karşıya bırakabilirdi. AdaGrad burada devreye girmekte ve adım boyutunu otomatik olarak belirlemektedir [14].

   Uyarlanabilir Gradyan Algoritması, gradyan tabanlı optimizasyon için bir algoritmadır. Öğrenme oranı, geçmiş gözlemleri hesaba dahil edilerek parametrelere bileşen olarak uyarlanır. Her parametrenin performansı artıran kendi öğrenme oranı vardır. NLP ve Görüntü tanıma konularında kullanımı uygundur.

4. Momentum: Momentum, Stokastik gradyan azalma ile kullanılan popüler bir tekniktir. Sadece mevcut durumda hesaplanan gradyan

değerini kullanmak yerine eski gradyanların değerlerini de kullanarak gideceği yönü belirlemeye çalışır.Salınım yaparak minimuma ulaşmaya çalışmaktadır. Çünkü bir batch için hatayı azaltacak olan parametre güncellemeleri verinin tamamında aynı etkiyi oluşturmayabilir [15].

5. RMSprop Optimizasyon (Root Mean Square Propagation): RMSprop optimizasyon, momentum optimizasyon ile benzerdir. RMSprop optimizasyon dikey olarak meydana gelen salınımı minimize eder. Dolayısıyla öğrenme oranını artırarak, yatay boyutta daha hızlı hareket edip minimuma daha hızlı ulaşma imkanı elde ederiz. RMSprop optimizasyon ve gradyan azalma arasındaki fark, matematiksel formülden kaynaklanmaktadır[15].

6. Adam Optimizasyonu (Adam Optimization): Adam, Adagrad ve RMSProp optimizatörlerinin bir kombinasyonudur[13]. Parametre güncellemelerini her bir parametre için ayrı ayrı ölçeklendirerek ve geçmiş gradyan bilgilerini kullanarak gradyan iniş optimizatörünün bir modifikasyonudur.Adam Optimizasyonu, gradyan iniş algoritmasının bir türevidir ve daha hızlı ve verimli bir şekilde çalışır. Bu algoritma, adaptif momentum ve adaptif öğrenme hızı kullanarak ağırlıkları günceller.

3.6 Aktivasyon Fonksiyonu

Yapay sinir ağlarına komplex verileri öğretebilmemiz için aktivasyon fonksiyonları gereklidir. Aktivasyon fonksiyonlarının amacı weight(ağırlık) ve bias değerlerini ayarlamaktır.
"Weight" (ağırlık), yapay sinir ağlarında girdi verileriyle çarpılarak nöronlara iletilen değerlerdir. Bu ağırlıklar, ağın öğrenme sürecinde

güncellenir ve çeşitli veri desenleri arasında ilişkileri öğrenmesine yardımcı olur. Ağırlıklar, ağın çıktılarını belirlemek için kullanılır ve doğru şekilde ayarlandığında, ağın istenen sonuçları üretmesine katkıda bulunurlar. "Bias" (yatay kayma), yapay sinir ağlarında her bir nöronun çıktısına eklenen sabit bir değerdir. Bu değer, nöronun aktivasyon fonksiyonunu değiştirerek ağın esnekliğini artırır. Bias, ağın veri setindeki karmaşıklıkları öğrenmesine ve girdi verileriyle daha iyi uyum sağlamasına yardımcı olur. Ayrıca, bias terimi, nöronların doğrusal olmayan ilişkileri öğrenmesine olanak tanır ve ağın genelleme yeteneğini artırabilir.

1. Sigmoid: Sigmoid, S şeklinde bir eğriye sahip bir aktivasyon fonksiyonudur. Giriş değerlerini 0 ile 1 arasında bir aralığa dönüştürür [16].

2. ReLU: ReLU (Rectified Linear Unit), derin öğrenme modellerinde kullanılan doğrusal olmayan bir aktivasyon fonksiyonudur. Girdi değeri pozitif ise, giriş değerini olduğu gibi çıktı olarak verir, aksi takdirde 0 değerini çıktı olarak verir. ReLU, vanishing gradient problemini çözmek ve modelin eğitimini hızlandırmak için yaygın olarak kullanılır [17].

3. Tanh: Hiperbolik tanjant fonksiyonudur. Sigmoid'e benzer, ancak -1 ve 1 arasında değerler üretir.

4. Softmax: Çoklu sınıflandırma problemleri için kullanılır.

3.7 Ses Türleri

- Konuşma (Metinden Sese)
- Müzik
- Müzik notaları (örnekler)
- Ses tasarımı

3.8 Ses Temsil Etme Yöntemi

- Raw-audio
- Spectrogramlar

3.8.1 Raw-audio

Bu ayar, standart .mp4 ses parçasına ek olarak videonuz için ayrı bir .wav dosyası oluşturur. Bu ayar, ayrı bir .wav dosyasının gelişmiş bir düzenleme programında paylaşmasını veya kullanmasını istiyorsanız kullanışlıdır. RAW ses parçasına uygulanacak işlem seviyesini seçebilirsiniz: Düşük: Minimum işleme uygular; post prodüksiyonda ses işleme uygularsanız idealdir. Orta: Manuel Ses Kontrolü ayarınıza (rüzgar ve / veya stereo) göre işleme uygular. Manuel Ses Kontrolünüz kapalıysa, kamera otomatik olarak rüzgar filtreleme ve stereo ses arasında geçiş yapar. Yüksek: Tam ses işleme (otomatik kazanç ve AAC kodlaması) uygular.

3.9 Spektrogram

Spektrogram, belirli bir dalga formunda bulunan çeşitli frekanslarda zaman içinde bir sinyalin sinyal gücünü veya "ses yüksekliğini" temsil etmenin görsel bir yoludur. Örneğin 2 Hz ile 10 Hz arasında daha fazla veya daha az enerji olup olmadığı görülebildiği gibi, enerji seviyelerinin zaman içinde nasıl değiştiği de görülebilir. Diğer bilim dallarında spektrogramlar genellikle insanlar, makineler, hayvanlar, balinalar, jetler vb. tarafından üretilen ve mikrofonlar tarafından kaydedilen ses dalgalarının frekanslarını görüntülemek için kullanılır. Sismik dünyada, spektrogramlar, farklı deprem türlerini veya yeryüzündeki diğer titreşimleri ayırt etmeye ve karakterize etmeye yardımcı olmak için bireysel veya sismometre grupları tarafından kaydedilen sürekli sinyallerin frekans içeriğine bakmak için giderek daha fazla kullanılmaktadır. [8]

3.10 Spektrum

Spektrum, elektromanyetik radyasyon dalgalarının sürekli bir aralığıdır. En uzun radyo dalgalarından en kısa X-ışınlarına ve gama ışınlarına kadar uzanır. Radyofrekans spektrumu, spektrumun alt kısmında yer alır [9].

3.11 Mel-Frekans Cepstral Katsayıları (MFCC-Mel-Frequency Cepstral Coefficients)

MFCC, ses sinyallerinin öznitelik çıkarma sürecinde kullanılan bir yöntemdir. Özellikle konuşma tanıma ve ses işleme alanlarında sıkça

kullanılır.

MFCC'nin kullanılmasının nedenleri arasında insan işitme sisteminin özelliklerini modellemeye uygunluğu, ses sinyallerinin temsilinin kompakt olması, ve gürültüye karşı dirençli olması gibi özellikler bulunur. Bu özellikler sayesinde, MFCC ses sinyallerinin analizi ve tanımlanması için etkili bir şekilde kullanılabilir. Konuşma tanıma bir denetimli öğrenme görevidir, belirli konuşma örnekleriyle eğitilerek konuşma seslerini tanıyabilen sistemler geliştirilir [10].

3.12 Üretimde Inputlar

- Conditioning, belirli koşullara bağlı olarak bir sistem veya modelin davranışının değiştirilmesidir.
- Autonomous, bir şeyin dış yardım veya kontrol olmaksızın kendi kendine hareket etme yeteneği veya özelliğidir.
- Continuation, sürecin veya durumun devam etmesini ifade eder.

3.13 Sinyaller;

Bir sinyalin dijitalleştirilmesi için gereken örnekleme, sinyalin zaman alanındaki temsilini belirler. Zaman alanı gösterimi, örneklendiği süre boyunca sinyalin genliklerini gösterir. Birçok uygulamada, sinyalin frekans bileşenleri hakkında ayrıntılı bilgi, sinyalin ve sinyali üreten sistemin daha iyi anlaşılması açısından önemlidir. Bir sinyalin frekans bileşenlerinin gösterimine frekans alanı gösterimi denir.

1-Dosya yükleme 2-Gerekliyse sinyalleri doldurma 3-Sinyallerden logaritmik spektrogramlarını çıkartma 4-Spectrogramları normalize etme
5-Normalleştirilmiş spektrogramların kaydedilmesi
->Preprocessing Pipeline [25]

3.13.1 Discrite Fourier Transform (DFT)

Ayrık Fourier dönüşümü (DFT) algoritması, sinyal örneklerini zaman alanından frekans alanına dönüştürür. DFT, spektral analiz, uygulamalı mekanik, akustik, tıbbi görüntüleme, sayısal analiz, enstrümantasyon ve telekomünikasyon alanlarında yaygın olarak kullanılmaktadır [20].

3.13.2 FFT

Bir sinyali bireysel spektral bileşenlere dönüştürerek sinyal hakkında frekans bilgisi sağlar. FFT'ler hata analizi, kalite kontrol ve makine veya sistemlerin durum takibi için kullanılır. FFT, (DFT) nin uygulanması için optimize edilmiş bir algoritmadır. Bir sinyal belirli bir zaman aralığında örneklenir ve frekans bileşenlerine ayrılır. Bu bileşenler, her biri kendi genlik ve fazına sahip farklı frekanslarda tek sinüzoidal salınımlardır[20].

3.13.3 Short Time Fourier Transform(STFT)

STFT, durağan olmayan sinyalleri analiz etmek için sinyal işlemede iyi bilinen bir tekniktir. STFT, sinyali dar zaman aralıklarına böler ve her bölümün Fourier dönüşümünü alır [22].

3.13.4 Değişkenleri

- Block size (Blok boyutu):FFT'nin hesaplanması için alınacak gerçek veri örneklerinin sayısını tanımlar.
- FFT size (FFT boyutu) : Sonuçta ortaya çıkan çizgilerin sayısını ve bununla birlikte gerçek ve sıfır dolgulu çizgiler arasındaki oranı tanımlar.
- Window type (Pencere türü) : Kullanılacak FFT penceresini açıklar. Eğitimde farklı pencere işlevlerinin kullanımına ilişkin iyi bir açıklama bulunmaktadır. Varsayılan olarak Blackman'ı kullanıyoruz çünkü bu, genlik hatası ile yan bantların genişliği arasında iyi bir uzlaşmadır.

- Overlap (Örtüşme) : İki FFT çekiminin birbiriyle ne kadar örtüşeceğini tanımlar. Kullanılan pencereden bağımsız olarak tüm örneklerin sonuçla aynı ağırlığa sahip olması için %50 yeterlidir [18].

3.14 Ses Normalizasyonu

- Peak Normalization
- Loudness Normalization

3.15 Peak Normalizasyonu

Ses sinyali, ses dalga biçiminde kaydedilen en yüksek noktaya referans verecek şekilde normalleştirilirse, bu, tepe normalleştirmesi olarak bilinir [24].
Sesinizin en duyulabilir bölümünü ele alalım; örneğimizde bu, birisinin bağırdığı bir ses kaydıdır.
Bu bağırışın sesini ölçtüğümüzü ve -6 dB olarak kaydedildiğini varsayalım.
Normalizasyon etkisini uyguladığımızda belli bir hesaplama devreye giriyor.
Burada amaç bağırılan kısmı güçlendirerek 0 dB düzeyine çıkarmaktır.
Bunu başarmak için bağırmanın mevcut ses yüksekliği ile bozulma olmadan ulaşabileceği maksimum ses seviyesi arasındaki eşitsizliği belirlememiz gerekir.
Basit bir ifadeyle (negatif sayılar nedeniyle ertelenmediğinizi varsayarak), hesaplama 0 dB'den -6 dB'nin çıkarılmasını içerir, bu da 6 dB'ye eşittir. Bu nedenle fark 6 dB'dir.

3.16 Loudness Normalizasyonu

Ses yüksekliği normalizasyonu kavramı, ortalama ses şiddetini hedef seviyeye getirmek için kazanç değerinin değiştirilmesidir.
Ortalama ses yüksekliği seviyesi, ortalama güç gibi basit bir ölçümden (örn. RMS), EBU R128 tarafından tanımlanan ve Replay Gain, Sound tarafından kullanılan gibi insan algısını dikkate alan daha hassas yöntemlere kadar farklı şekillerde tahmin edilebilir. Çek ve GoldWave.
Örneğin YouTube, ses içeriklerinin ses yüksekliğinin -14 LUFS olmasını tercih ediyor. Bir ses programı analiz edilirse ve -10 LUFS'de bulunursa YouTube, ses yüksekliğini 4 dB azaltarak tercih edilen seviyeye getirecek. Ses yüksekliği normalleştirmesi, birden fazla şarkıyı art arda çalarken ses yüksekliği seviyelerinin değişmesi sorununu giderir. Ses yüksekliğinin normalleştirilmesinden önce, çalma listesindeki bir şarkı diğerlerinden daha sessiz olabilir ve bu da dinleyicinin ses düzeyini tekrar tekrar ayarlamasına neden olabilir.

3.17 Evidence Lower Bound(ELBO)

Evidence Lower Bound (ELBO), özellikle varyasyonel bayes yöntemlerinde kullanılan bir optimizasyon kriteridir[29] [26]. Temel amacı, karmaşık olasılık dağılımlarını tahmin etmek ve optimize etmektir[28] [30]. ELBO, bir hedef olasılık dağılımına en yakın parametreli bir dağılım bulmak için kullanılır ve şu şekilde ifade edilir:

$$\text{ELBO}(q) = \mathbb{E}_{q(z)}[\log p(x, z) - \log q(z)][27] \tag{12}$$

Burada (12):

- $q(z)$: Yaklaşık dağılım (variational distribution)
- $p(x, z)$: Birlikte olasılık (joint probability) dağılımı
- x: Gözlemler
- z: Gizli değişkenler (latent variables)

ELBO, gözlemlenen verilerin marjinal olasılığının (marjinal likelihood) bir alt sınırıdır. Maksimize edilmesi, hedef dağılım $p(z|x)$'ye yakınsayan bir $q(z)$ yaklaşık dağılımı bulmamızı sağlar.
ELBO, varyasyonel bayes yöntemlerinde kullanılır ve aşağıdaki iki amacı yerine getirmek için optimize edilir:

1. Yaklaşık dağılım $q(z)$'nin hedef dağılım $p(z|x)$'ye yakın olması.
2. Gözlemlenen veri x'nin marjinal olasılığının (evidence) optimize edilmesi.

ELBO'nun Avantajları;

1. **Karmaşık Modellerde Kullanılabilirlik**: ELBO, karmaşık olasılık modellerini optimize etmek için kullanılabilir, bu da doğrudan hesaplanması zor olan integralleri yaklaştırmak için uygundur.

2. **Stokastik Gradient Descent (SGD) ile Uyumlu**: ELBO, SGD gibi optimizasyon yöntemleriyle kolayca optimize edilebilir.

3. **Skalabilite**: Büyük veri kümeleri ve yüksek boyutlu gizli değişkenlerle başa çıkabilir.

ELBO'nun Dezavantajları;

1. **Yaklaşım Hataları**: Yaklaşık dağılım $q(z)$'nin, hedef dağılım $p(z|x)$ ile tamamen örtüşmemesi durumunda, sonuçlar suboptimal olabilir.

2. **Rekabetçi Alternatifler**: Markov Chain Monte Carlo (MCMC) gibi yöntemler, özellikle doğruluğun kritik olduğu durumlarda daha iyi sonuçlar verebilir.

3. **Variational Gap**: ELBO, marjinal olasılığı her zaman alttan sınırlar ve bu sınır farkı (gap) bazı durumlarda önemli olabilir.

ELBO yerine bazı durumlarda kullanılan kavramlar şunlardır:

1. **Variational Free Energy (VFE)**: Genellikle ELBO ile aynı anlamda kullanılır ve enerjinin minimize edilmesi perspektifinden ifade edilir.

2. **Negative Log Evidence (NLE)**: ELBO'nun negatif hali olup, optimize edilirken minimize edilmesi gereken bir kayıp fonksiyonu olarak ele alınır.

3. **Variational Objective**: Varyasyonel bayes yöntemlerinin genel hedef fonksiyonlarını ifade etmek için kullanılır.

4 Bulgu ve Tartışma

4.1 Hatalar

```
Total params: 2,371,777 (9.05 MB)
Trainable params: 2,368,833 (9.04 MB)
Non-trainable params: 2,944 (11.50 KB)
Traceback (most recent call last):
File "C:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\genera
networks-main\n14 Sound generation with VAE\code\train.py", line 41, in <module
autoencoder = train(data, LEARNING_RATE, BATCH_SIZE, EPOCHS)
File "C:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\genera
networks-main\n14 Sound generation with VAE\code\train.py", line 36, in train
autoencoder.train(train_dataset, batch_size, epochs)
File "C:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\genera
networks-main\n14 Sound generation with VAE\code\autoencoder.py", line 59, in t
File "C:\Users\dgdl\AppData\Local\Programs\Python\Python312\Lib\site-packages\k
data_adapters\__init__.py", line 124, in raise_unsupported_arg
raise ValueError(
ValueError: When providing ‘x‘ as a tf.data.Dataset, ‘y‘ should not be passed.
 targets should be included as part of the tf.data.Dataset.
```

```
Traceback (most recent call last):
File "c:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\gener
```

```
autoencoder = train(None, LEARNING_RATE, BATCH_SIZE, EPOCHS)
File "c:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\gener
autoencoder.train(train_dataset, batch_size, epochs)
File "c:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\gener
self.model.fit(x_train,
File "C:\Users\dgdl\AppData\Local\Programs\Python\Python312\Lib\site-packages\
raise e.with_traceback(filtered_tb) from None
File "C:\Users\dgdl\AppData\Local\Programs\Python\Python312\Lib\site-packages\
raise ValueError(
ValueError: When providing `x` as a tf.data.Dataset, `y` should not be passed.
PS C:\Users\dgdl\Downloads\generating-sound-with-neural-networks-main\generati
2024-05-09 06:17:40.236300: I tensorflow/core/platform/cpu_feature_guard.cc:11
2024-05-09 06:17:40.979272: I tensorflow/core/platform/cpu_feature_guard.cc:11
WARNING:tensorflow:From c:\Users\dgdl\Downloads\generating-sound-with-neural-n
```

```
Trainable params: 2,368,833 (9.04 MB) Non-trainable params: 2,944 (11.50 KB) Ep
Traceback (most recent call last):
File
"c:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generating
n\14 Sound generation with VAE\code\train.py", line 45, in <module>
autoencoder = train(x_train, LEARNING RATE, BATCH_SIZE, EPOCHS)
File
```

```
"c:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generating
n\14 Sound generation with VAE\code\train.py", line 39, in train
autoencoder.train(x_train, batch_size, epochs)
File
"c:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generating
n\14 Sound generation with VAE\code\autoencoder.py", line 59, in train
self.model.fit(x_train,
File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages\
File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages\
ValueError: Exception encountered when calling Functional.call().
Invalid input shape for input Tensor("data:0", shape=(64, 1), dtype-float32). E
Arguments received by Functional.call():
inputs tf.Tensor(shape=(64, 1), dtype-float32)
training=True
• mask=None
PS C:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generati


 PS C:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generat
 File
 "c:\Users\dgdln\Downloads\generating-sound-with-neural-networks-main\generatin
 n\14 Sound generation with VAE\code\preprocess.py", line 13, in <module>
 import librosa
 File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages
 File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages

 File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages
 File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages
```

```
File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages
File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-packages
```

4.2 epsilon

```
Traceback (most recent cali last) :
File "C:\Users\dgdln\Desktop\generating-sound-with-neural-
networks-main\14SoundGenerationwithVAE\code\train.py",
line 5, in «module»
from autoencoder import VAE
File "C:\Users\dgdln\Desktop\generating-sound-with-neural
-networks-main\14SoundGenerationwithVAE\code\autoencoder.py"
, line 15, in «module»
epsilon = RandoNormalLayer()([self.mu, self. logvar])
NameError: name 'RandoNormalLayer' is not defined
```

4.3 k shape

```
Epoch 1/150
Traceback (most recent call last) :
File"C:\Users\dgdln\Desktop\generating-sound-with-neural-
networks-main\14SoundGenerationwithVAE\code\train.py",
```

```
line 88, in <module>
autoencoder = train(train_dataset, LEARNING_RATE, EPOCHS)

File"C:\Users\dgdln\Desktop\generating-sound-with-neural-
networks-main\14SoundGenerationwithVAE\code\train.py",
line 82, in train
autoencoder.model.fit(train_dataset, epochs=epochs, steps_per_epoch=
num_batchs,shuffle=True)

File "C:Users/dgdln/AppData/Local/Programs/Python/Python312/Lib/

site-packages/keras/src/utils/traceback_utls.py", line 122, in
error_handler

raise e.with_traceback(filtered_tb) from None

File "C:\Users\dgdln\Desktop\generating-sound-with-neural-networks-

main\14SoundGenerationwithVAE\code\autoencoder.py", line 238,
in sample_point_from_normal_distribution
epsilon=K.random_normal(shape=tf.convert_to_tensor(K.shape(
self.mu),dtype=tf.float32),mean=0.,stddev=1.)#K.shape(self.mu),
mean=0.,stddev=1.)

ValueError: Exception encountered when calling Lambda.call().
```

4.4 model fit

```
Traceback (most recent call last) :
File"C:\Users\dgdln\Desktop\generating-sound-with-neural-networks-
main\14SoundGenerationwithVAE\code\train.py", line 44, in «module»
File"C:\Users\dgdln\Desktop\generating-sound-with-neural-networks-
main\14SoundGenerationwithVAE\code\train.py", line 59, in train
self. model. fit (x_train,
File  "C:Users/dgdln/AppData/Local/Programs/Python/Python312/Lib/site-
packages/keras/src/utils/traceback_utls.py", line 122, in error_handler
raise tb) from None
raise e.with_traceback(filtered_tb) from None
File  "C:Users/dgdln/AppData/Local/Programs/Python/Python312/Lib/site-
packages/tensorflow/python/data/ops/dataset_ops.py" line 503 , in__iter__
raise RuntimeError("'tf.data.Dataset' only supports Python_style"
RuntimeError: 'tf.data.Dataset' only supports Python-style iteration in
eager mode or within tf.function.
```

4.5 tensorflow function hatası

```
ValueError: Exception encountered when calling Lambda.call() .
A KerasTensor cannot be used as input to a TensorFlow function.
A KerasTensor is a symbolic placeholder for a shape and dtype,
used when constructing Keras Functional models or Keras Functions.
You can only use it as input to a Keras layer or a Keras operation
(from the namespaces'keras.layers' and 'keras.operations'). You are
likely doing something like:
```

```
...
x Input(...)
...
tf_fn(x) # Invalid.
...
What you should do instead is wrap 'tf_fn' in a layer:
...
class MyLayer(Layer) :
def call(self, x):
return tf_fn(x)

x=MyLayer()(x)
...
Arguments received by Lambda.call():
• inputs=[ 'tf.Tensor(shape=(None,128), dtype=float32) ']
• mask-[ 'None' ,' None ' ]
• training=True
```

4.6 y target

```
Traceback (most recent call last):
File "c:\users\dgdin\Downloads\generating-sound-with-neural-networks-main
\generating-sound-with-neural-networks-main\generating-sound-with-
neural-networks-mai
n\14 Sound generation with VAE\code\train.py", line 77, in <module>
autoencoder = train(None, LEARNING RATE, BATCH_SIZE, EPOCHS)
File "c:\Users\dgdin\Downloads\generating-sound-with-neural-networks-main
```

```
\generating-sound-with-neural-networks-main\generating-sound-with-
neural-networks-mai
n\14 Sound generation with VAE\code\train.py", line 73, in train
autoencoder.train(train dataset, batch size, epochs)
File "c:\Users\dgdIn\Downloads\generating-sound-with-neural-networks-main
\generating-sound-with-neural-networks-main\generating-sound-with-
neural-networks-mai
n\14 Sound generation with VAE\code\autoencoder.py", line 59, in train
self.model.fit(x_train,
File "C: \users\dgdIn\AppData\Local\Programs\Python\Python212\Lib\site-
packages\keras\src\utils\traceback_utils.py", line 122, in error_handler
raise e.with_traceback(filtered_tb) from None
File "C: \users\dgdIn\AppData\Local\Programs\python\Python312\Lib\site-
packages\keras\src\trainers\data_adapters\_init__.py", line 124,
in raise_unsupported_arg
raise valueerror(
ValueError: hen providing ‘x’ as a tf.data.Dataset, "y® should not
be passed. Instead, the targets should be included as part of the
tf.data.Dataset
PS C:\Users\dgdln\Downloads\generating-sound-with-neural-networks-
main\generating-sound-with-neural-networks-main\generating-sound
-with-neural-networks-main\14sound generation with VAE\code>
& C:/Users/dgdln/AppData/Local/Programs/Python/Python312
/python.exe "c:/Users/dgdln/Downloads/generating-sound-with-
neural-networks-main/generating-sound-with-neural-networks-main/
generating-sound-with-neural-networks-main/14Soundgenerationwith VAE/
code/train.py"
```

```
2024-05-09 06:17:49.236398: I tensorflow/core/util/port.cc:113]

oneDNN custom operations are on. You may see slightly different numerical
results due to floatin

g-point round-off errors from different computation orders.
To turn them off, set the environment variable "TF_ENABLE_ONEDNN_OPTS=0" .
2024-05-09 06:17:49.979272: I tensorflow/core/util/port.cc:113] oneDNN
custom operations are on. You may see slightly different numerical
results due to floatin  g-point round-off errors from
different computation orders.
To turn them off, set the environment variable "TF_ENABLE_ONEDNN_OPTS=0" .
WARNING:tensorflow: From c: \Users\dgdln\Downloads\generating-
sound-with-neural-
networks-main\generating-sound-with-neural-networks-main
\generating-sound-with-neural-networks-main\14
Soundgenerationwith VAE\code\autoencoder.py:14: The
name tf.disable_eager_execution is deprecated.
Please use tf.compat.v1.disable eager_
execution instead.
```

4.7 SymbolicTensor

```
Total params: 2,371, 777 (9.05 MB)
Trainable params:  (9.04 MB)
Non-trainable params: 2,944 (11.50 KB)
Traceback (most recent call last) :
```

```
File"C:\Users\dgdln\Desktop\generating-sound-with-neural-networks

-main\14SoundGenerationwithVAE\code\train.py", line 87, in <module>
autoencoder = train(train_dataset, LEARNING_RATE, EPOCHS)

File "C:\Users\dgdln\Desktop\generating-sound-with-neural-networks
-main\14SoundGenerationwithVAE\code\train.py", line 78, in train
num_batches = tf.data.experimental.cardinality(train_dataset) .numpy()

File "C:\Users\dgdln\AppData\Local\Programs\Python\Python312\Lib\site-
packages\tensorfIow\python\framework\tensor.py", line 260, in _getattr
self ._getattribute (name)
AttributeError: 'SymbolicTensor' object has no attribute 'numpy '
```

4.8 Hata1

```
map data = lambda filename: tf.compat.v1.py_func
(load, [filename], [tf.float32])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.10/
dist-packages/tensorflow/python/autograph/impl/api.py:460:
py_func (from tensorflow.python.ops.script_ops) is deprecated
and will be removed
in a future version. Instructions for updating:
```

tf.py_func is deprecated in TF V2. Instead, there are two options available in V2.
- tf.py_function takes a python function which manipulates tf eager tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
an ndarray (just call tensor.numpy()). tf.py_function also takes tf eager
tensors as inputs by using tf.Tensor objects as well as being differentiable using a gradient tape.
- tf.numpy_function maintains the semantics of the deprecated tf.py_func (it is not differentiable, and manipulates numpy arrays).
It drops the stateful argument making all functions stateful.

Solution:
map_data = lambda filename: tf.py_function(func=load, inp=[filename], Tout=tf.float32)

4.9 Hata2

```
# Pick a sample of the test set for generating output images
assert BATCH_SIZE >= num_examples_to_generate
for test_batch in test_dataset.take(1):
test_sample = test_batch[0]


---------------------------------------------------------------
InvalidArgumentError
Traceback (most recent call last)
<ipython-input-35-0120a0afdd31> in <cell line: 3>()
1 # Pick a sample of the test set for generating
output images
2 assert BATCH_SIZE >= num_examples_to_generate
----> 3 for test_batch in test_dataset.take(1):
4     test_sample = test_batch[0]

3 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/
python/data/ops/iterator_ops.py in _next_internal(self)
808   def _next_internal(self):
809     try:
--> 810       return self._next_internal()
811     except errors.OutOfRangeError:
812       raise StopIteration

/usr/local/lib/python3.10/dist-packages/tensorflow/
```

```
python/data/ops/iterator_ops.py in _next_internal(self)
771 # try to communicate that there is no more dat
a to iterate over.
772 with context.execution_mode(context.SYNC):
--> 773   ret = gen_dataset_ops.iterator_get_next(
774 self._iterator_resource,
775 output_types=self._flat_output_types,

/usr/local/lib/python3.10/dist-packages/tensorflow/
python/eager/context.py in execution_mode(mode):
3027 return _result
3028 except core._NoGradientException as e:
-> 3029 raise_from_not_ok_status(e, name)
3030 except core._FallbackException:
3031 pass

/usr/local/lib/python3.10/dist-packages/tensorflow/python/
framework/ops.py in raise_from_not_ok_status(e, name)
5881 def raise_from_not_ok_status(e, name):
5882   e.message += ("; name: " + str(name) if name is
not None else "")
-> 5883   raise core._status_to_exception(e) from None
# pylint: disable=protected-access
5884
5885

InvalidArgumentError: {{function node __wrapped__
```

```
  IteratorGetNext_output_types_1_device_/job:localhost/
  replica:0/task:0/device:CPU:0}} TypeError: Invalid file:
<tf.Tensor: shape=(), dtype=string, numpy=b'Data/
gunshot_audio_dataset_archive//1 (21).wav'>
Traceback (most recent call last):

File "/usr/local/lib/python3.10/dist-packages/tensorflow/
python/ops/script_ops.py", line 268, in __call__
return func(device, token, args)

File "/usr/local/lib/python3.10/dist-packages/tensorflow/
python/ops/script_ops.py", line 146, in __call__
outputs = self._call(device, token, args)

File "/usr/local/lib/python3.10/dist-packages/tensorflow/
python/ops/script_ops.py", line 153, in _call
ret = self._func(*args)

File "/usr/local/lib/python3.10/dist-packages/tensorflow/
python/autograph/impl/api.py", line 643, in wrapper
return func(*args, **kwargs)

File "<ipython-input-22-96bfb49b52d4>", line 13, in load
data, sampling_rate = librosa.load(file_, sr=sr,
offset=0.0, duration=duration)

File "/usr/local/lib/python3.10/dist-packages/librosa/
```

```
core/audio.py", line 176, in load
y, sr_native = __soundfile_load(path, offset,
duration, dtype)

File "/usr/local/lib/python3.10/dist-packages/librosa/core/
audio.py", line 209, in __soundfile_load
context = sf.SoundFile(path)

File "/usr/local/lib/python3.10/dist-packages/soundfile.py",
line 658, in __init__
self._file = self._open(file, mode_int, closefd)

File "/usr/local/lib/python3.10/dist-packages/soundfile.py",
line 1212, in _open
raise TypeError("Invalid file: {!r}".format(self.name))

TypeError: Invalid file: <tf.Tensor: shape=(), dtype=string,
numpy=b'Data/gunshot_audio_dataset_archive/AK-47/1 (21).wav'>
```

4.10 Hata3

```
generate_and_save_images(model, 0, test_sample, 'AK-47')



ValueError Traceback (most recent call last)
<ipython-input-50-f4472f6991b> in <cell line: 1>()
----> 1 generate_and_save_images(model, 0, test_sample,
'AK-47')

<ipython-input-48-05cce7f0ecf5>
in generate_and_save_images(model, epoch, test_sample, save)
9 for i in range(predictions.shape[0]):
---> 10 plt.subplot(4, 4, i + 1)
11 wave = np.asarray(predictions[i])
12 #librosa.display.waveplot(wave[0], sr=3000)

/usr/local/lib/python3.10/dist-packages/matplotlib/
pyplot.py in subplot(*args, **kwargs)
1321 # First, search for an existing subplot with
a matching spec.
1322 key = SubplotSpec._from_subplot_args(fig, args)
-> 1323 bbox = ax.get_position()
1324 for ax in fig.axes:
1325 ax._request_position(fig.transFigure.inverted().
transform_bbox(bbox))
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/gridspec.py
in _from_subplot_args(figure, args)
596 else:
597 if not isinstance(num, Integral) or
num < 1 or num > rows*cols:
--> 598 raise ValueError(f"num must be an integer
with 1 <= num <= {rows*cols}, not {num!r}")
599 return self[numpy.floor(num).astype(int) - 1]
600

ValueError: num must be an integer with 1 <= num <= 16, not 17
```

5 Sonuç

Bu projede, üretici ağlar kullanılarak ses üretmek için üç farklı çalışma incelendi. İlk olarak OpenAI Jukebox projesi, ardından Valerio Velardo'nun "The Sound of AI" YouTube kanalındaki proje ve son olarak Kaggle üzerindeki müzik üretim projesi ele alındı.
Projenin ilk haftasında, VQ-VAE (Vector Quantized-Variational Autoencoder) gibi yapay zeka teknikleri kullanılarak farklı frekanslarda sesler üretebilmek amacıyla yapılan kod inceleme dönemindeki karşılaşılan bulgular nedeniyle aynı dönemde ikinci kaynak kullanılmaya başlanmış, ilerleme kaydedilmiştir.
Proje kapsamında yapılan çalışmalar sonucunda şunlar elde edildi: Spektrogramlaştırma, sinyal doldurma, ses normalleştirmesi ve eğitim

sırasındaki spektrogram grafikleri gibi ses işleme teknikleri kullanıldı. Farklı ses datasetleri üzerinde çeşitli kod projeleri yönetildi. Verilen hatalar çözüldü ve eğitimler sonucunda bazı çalışmalar tamamlandı.
Öneriler Gelecekteki çalışmalar şu şekilde ilerletilebilir: Bulgular çözümlenerek farklı projeler sonucundaki datasetin çıktılarının karşılaştırılabilinir. Daha fazla ses dataseti üzerinde çalışmalar yapılabilir. Farklı yapay zeka teknikleri kullanılabilir.

Kaynakça

[1] Author/OpenAI, "Müzik kutusu - jukebox (openai)." https://openai.com/research/jukebox/, Year Published/ 02.2024. Accessed: 15.03.2024.

[2] A. Gezer, "Vae." https://www.yapayzekatr.com/2023/11/14/variational-autoencoders-vaes-nedir/, Year Published/ 14.11.2023. Accessed: 15.03.2024.

[3] A. Kumar, "A crash course on vaes, vq-vaes, and vae-gans." https://ammesatyajit.medium.com/a-crash-course-on-vaes-vq-vaes-and-vae-gans-3fdcc40b059e, Year Published/ 29.01.2020. Accessed: 17.03.2024.

[4] A. Yadav, "Understanding vector quantized variational autoencoders (vq-vae)." https://shashank7-iitd.medium.com/understanding-vector-quantized-variational-autoencoders-vq-vae-323d710a888a, Year Published/ 01.09.2019. Accessed: 17.03.2024.

[5] A. Mishra, "An introduction to vae-gans." https://wandb.ai/shambhavicodes/vae-gan/reports/An-Introduction-to-VAE-GANs--VmlldzoxMTcxMjM5, Year Published/ 08.12.2021. Accessed: 17.03.2024.

[6] Engel, "Nsynth." https://paperswithcode.com/dataset/nsynth/, 01.04.2017. 25.03.2024.

[7] J. Vial, "The maestro dataset v2." https://www.kaggle.com/datasets/jackvial/themaestrodatasetv2/, 01.01.2018. 25.03.2024.

[8] PNSN, "What is a spectrogram?." https://pnsn.org/spectrograms/what-is-a-spectrogram/, 01.01.2020. 28.03.2024.

[9] ACMA, "What is spectrum?." https://www.acma.gov.au/what-spectrum#:~:text=The%20spectrum%20is%20a%20continuous,Frequency%20bands/, 17.10.2022. 28.03.2024.

[10] U. Kiran, "Mfcc technique for speech recognition." https://community.gopro.com/s/article/What-is-RAW-Format?language=en_US/, 14.08.2023. 28.03.2024.

[11] Author/Emre Yüksel, "Entropi&KL Divergence." https://medium.com/kaveai/d%C3%BCzensizlik-entropy-%C3%A7apraz-d%C3%BCzensizlik-cross-entropy-ve-kl-iraksakl%C4%B1%C4%9F%C4%B1-kl-divergence-89d26735789f/, Year Published/ 02.2024. Accessed: 15.03.2024.

[12] Author/Unreal Engine, "UE BP." https://dev.epicgames.com/documentation/en-us/unreal-engine/making-macros-in-unreal-engine?application_version=5.0/, Year Published/ 02.2024. Accessed: 02.03.2024.

[13] A. O. YILMAZ, "Derin Öğrenmede kayıp fonksiyonları ve optimizasyon algoritmaları." https://aoyilmaz.medium.com/derin-%C3%B6%C4%9Frenmede-kay%C4%B1p-fonksiyonlar%C4%B1-ve-optimizasyon-algoritmalar%C4%B1-8fc754e7a639/, Year Published/ 27.07.2023. Accessed: 25.03.2024.

[14] Author/wikipedia, "Stochastic gradient descent." https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad/. Accessed: 25.03.2024.

[15] A. ATCILI, "Yapay sinir aglarinda kullanilan optimizasyon algoritmalari." https://medium.com/machine-learning-t%C3%BCrkiye/yapay-sinir-a%C4%9Flar%C4%B1nda-kullan%C4%B1lan-optimizasyon-algoritmalar%C4%B1-3e87cd738cb5/, Year Published/ 06.11.2020. Accessed: 25.03.2024.

[16] Author/wikipedia, "Sigmoid function." https://en.wikipedia.org/wiki/Sigmoid_function/. Accessed: 25.03.2024.

[17] Author/wikipedia, "Rectifier (neural networks)." https://en.wikipedia.org/wiki/Rectifier_(neural_networks)/. Accessed: 25.03.2024.

[18] Ron Fredericks, *Short-time Fourier Transform*, https://www.biophysicslab.com/2020/07/10/short-time-fourier-transform/, July 10, 2020, [Online; accessed 01.05.2024].

[19] David Munson and Andrew Singer, *06 - Short Time Fourier Transform*, https://www.youtube.com/watch?v=TZzS52OplYs, May 20, 2010, [Online; accessed 01.05.2024].

[20] NTI Audio, *Fast Fourier Transformation FFT - Basics*, https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft, 2024, [Online; accessed 01.05.2024].

[21] University of Wisconsin–Madison, *Limitations of the Fourier Transform: STFT*, https://qiml.radiology.wisc.edu/wp-content/

uploads/sites/760/2020/10/notes_016_stft.pdf, 2024, [Online; accessed 01.05.2024].

[22] Carnegie Mellon University, *3. Short-Time Fourier Transforms*, https://course.ece.cmu.edu/~ece491/lectures/L25/STFT_Notes_ADSP.pdf, 2024, [Online; accessed 01.05.2024].

[23] R.W.A. Scarr, *Normalization and adaptation of speech data for automatic speech recognition*, https://www.sciencedirect.com/science/article/abs/pii/S0020737370800207, January 1970, [Online; accessed 01.05.2024].

[24] Poudel Priyanka, *Audio Normalization*, https://medium.com/@poudelnipriyanka/audio-normalization-9dbcedfefcc0, Sep 5, 2023, [Online; accessed 01.05.2024].

[25] Valerio Valerdo, *generating-sound-with-neural-networks*, https://github.com/musikalkemist/generating-sound-with-neural-networks/blob/main/12%20Preprocessing%20pipeline/preprocess.py, 2021, [Online; accessed 01.05.2024].

[26] Matthew N. Bernstein, *The evidence lower bound (ELBO)*, https://mbernste.github.io/posts/elbo/, May 25, 2020, [Online; accessed 18.05.2024].

[27] Sunshine, *Evidence Lower Bound(ELBO)*, https://medium.com/@sunshine990316/evidence-lower-bound-elbo-63ec2e4c1b38, May 25, 2023, [Online; accessed 17.05.2024].

[28] Yunfan's Blog, *ELBO — What & Why*, https://yunfanj.com/blog/2021/01/11/ELBO.html, Jan 11, 2021, [Online; accessed 17.05.2024].

[29] Hesham Ali, *What is Evidence Lower Bound (ELBO)?*, https://mlarchive.com/deep-learning/evidence-lower-bound/, Jan 1, 2021, [Online; accessed 17.05.2024].

[30] Machine Learning & Simulation, *Variational Inference | Evidence Lower Bound (ELBO) | Intuition & Visualization*, https://www.youtube.com/watch?v=HxQ94L8n0vU, May 11, 2021, [Online; accessed 17.05.2024].

[31] Kaggle, *FSDD*, https://paperswithcode.com/dataset/fsdd, Apr 27, 2020, [Online; accessed 17.05.2024].

[32] Kaggle, *Gun Shot*, https://www.kaggle.com/datasets/emrahaydemr/gunshot-audio-dataset, Jan 1, 2021, [Online; accessed 01.05.2024].

[33] kCode, *Generate music with Variational AutoEncoder*, https://www.kaggle.com/code/basu369victor/generate-music-with-variational-autoencoder, Jan 1, 2021, [Online; accessed 01.05.2022].