İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ SİSTEMLERDE SESLİ KOMUT TANIMA

YÜKSEK LİSANS TEZİ

Can ÇETİN

Mekatronik Mühendisliği Ana Bilim Dalı

Mekatronik Mühendisliği Programı

HAZİRAN 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ SİSTEMLERDE SESLİ KOMUT TANIMA

YÜKSEK LİSANS TEZİ

Can ÇETİN
(518171007)

Mekatronik Mühendisliği Ana Bilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Mustafa DOĞAN

HAZİRAN 2020

İTÜ, Fen Bilimleri Enstitüsü'nün 518171007 numaralı Yüksek Lisans Öğrencisi Can ÇETİN'in, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "Gömülü Sistemlerde Komut Tanıma" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Mustafa Doğan** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Dr. Ögr. Üyesi Ali Fuat Ergenç** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Dr. Öğr. Üyesi Ebru Arısoy Saraçlar** Mef Üniversitesi	.....

Teslim Tarihi : 22 Haziran 2020
Savunma Tarihi : 06 Temmuz 2020

ÖNSÖZ

Tez çalışmamda benden yardımlarını esirgemeyen ve çalışmalarımda yol haritamı çizen Sayın hocam Prof. Dr. Mustafa Doğan'a; tezime yaptıkları değerli katkılardan dolayı sayın jüri üyelerim Dr. Öğr. Üyesi Ali Fuat Ergenç'e ve Dr. Öğr. Üyesi Ebru Arısoy Saraçlar'a; her türlü maddi ve manevi desteği ile yanımızda olan Merkez Ar&Ge Elektronik Sistemler Grup Yöneticimiz Sayın Onur Dinçer'e; çalışmalarım sırasında bana sağladığı imkanlar, maddi ve manevi verdiği desteği ile her zaman arkamda duran, beni dinleyen ve problemlerime çözüm bulan, her zaman pozitif gülümsemesi ile desteğini asla esirgemeyen Arçelik Merkez Ar&Ge Gömülü Sistemler Teknoloji Aile Lideri Sayın Çağatay Büyüktopçu'ya; Yaşadığım bütün sıkıntılarda yanımda olan NXP Semiconductors firmasından Sayın Paul Le Boulenger'a ve Sayın Taylan Mucuk'a; Deep Learning alanında benimle bilgisini paylaşan ve yardımlarını esirgemeyen Arçelik Gömülü Sistemler Teknoloji Ailesinden çok sevdiğim abilerim Sayın Kıdemli Uzman Yüksek Mühendis Özkan Akgül'e ve Kıdemli Uzman Mühendis Sayın Özgür Özkan'a; Elektronik donanım konusunda bana sağladığı değerli desteği için Arçelik Gömülü Sistemler Teknoloji Ailesinden Sayın Uzman Mühendis Görkem Savaş'a; ve son olarak beni bu yaşıma kadar getiren her anımda yanımda olan canım aileme sonsuz teşekkür ve şükranlarımı borç bilirim. Bu tez kapsamında Arçelik A.Ş Merkez Arge çalışmaları kapsamında gerçekleştirilmiştir.

İyi ki varsınız….

Haziran 2020

Can ÇETİN
Elektrik Elektronik Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ……v
İÇİNDEKİLER……vii
KISALTMALAR……ix
ÇİZELGE LİSTESİ……xi
ŞEKİL LİSTESİ……xiii
ÖZET……xvii
SUMMARY……xix
1. GİRİŞ……1
1.1 Literatür Taraması……1
1.2 Konuşma Tanımada Temel Kavramlar……3
1.3 Konuşma Tanıma Kullanım Alanları……4
1.4 Konuşma Tanıma Sistemlerinin Sınıflandırılması……5
1.4.1 Sesin sürekliliğine göre……5
1.4.1.1 Ayrık kelimeler……5
1.4.1.2 Sürekli konuşma……5
1.4.1.3 Bağlı kelimeler……5
1.4.1.4 Doğal konuşma……6
1.4.2 Konuşmacıya bağımlılığına göre……6
1.4.2.1 Kişiye bağımlı ……6
1.4.2.2 Ayrık kelimeler……6
1.4.3 Temel alınan birime göre……6
1.4.3.1 Sözcük tabanlı……6
1.4.3.2 Fonem tabanlı……6
1.4.4 Metin Bağımlılığına göre……7
1.4.4.1 Metne dayalı tanıma……7
1.4.4.2 Metinden bağımsız tanıma…… 7
1.5 Kurulacak Modele Karar Verilmesi……7
1.6 Konuşma Tanıma Sistemlerinde Temel Sorunlar……7
1.7 Eko Engelleme (Echo Cancellation)……8
1.8 Gürültü Bastırma (Noise Suppression)……9
1.8.1 Hüzme biçimlendirme (Beamforming)……9
1.9 Tez Organizasyonu……10
2. SİNYAL İŞLEME……13
2.1 Sayısal Sinyallerin İşlenmesi……13
2.1.1 MFCC öznitelik çıkarım yöntemi……13
2.1.2 Kullanılan programlar……23
2.1.2.1 PC'de sinyal işleme ortamları……23
2.1.2.2 Gömülü sistemlerde ses işleme ortamı……27
2.2 Ses İşleme Çıktıları……27

3. DERİN ÖĞRENME……..29
3.1 Derin Öğrenmeye Giriş…….29
3.1.1 Supervised learning'e giriş…….33
3.1.1.1 Yapay sinir ağları…….34
3.1.1.2 Yapay sinir ağlarının eğitilmesi ve sınıflandırması.....34
3.1.1.2.1 Feed forward neural networks.....42
3.1.1.2.2 Convolutional neural network.....42
3.1.1.2.3 Recurrent neural networks.....45
3.2 Kullanılan Yapay Zekâ Kütüphaneleri…….49
3.2.1 Yapay zeka kütüphanesi olarak: TensorFlow...50
3.2.1.1.1 Keras kütüphanesinin TensorFlow ile kullanılması.....52
3.2.1.1.2 TensorFlow çalışması sonucu elde edilen çıktılar.....60
4. GÖMÜLÜ SİSTEMLER MİMARİSİ TASARIMI……..……61
4.1 Tez Kapsamında Kullanılan İşlemci Mimarisi……...61
4.2 Hardware Mimarisi……..65
4.2.1 Mikrofon……66
4.2.2 Codec yapısı ve HW mimarisinde kullanımı……...68
4.2.3 ADC yapısı ve HW mimarisindeki yeri……..71
4.2.4 DAI HW mimarisindeki yeri……...73
4.2.5 HW mimari çıktıları…….77
4.3 SW Mimarisi……77
4.3.1 Driver katmanı……..79
4.3.2 MFCC hesaplamaları……80
4.3.3 TFLite deep learning model……80
4.3.4 Wakeup word detection…..82
4.3.5 KWS (Key word spotting)……82
4.3.6 SW mimarisi çıktıları……83
5. DENEYSEL SONUÇLAR ve TARTIŞMA……85
6. SONUÇ VE ÖNERİLER……..95
KAYNAKLAR …..97
EKLER.....101
ÖZGEÇMİŞ……..105

KISALTMALAR

ANN: Yapay Sinir Ağları (Artifical Neural Network)

CNN: Evrişimsel Sinir Ağları (Convolutional Neural Network)

DCT: Ayrık Kosimüs Dönüşümü (Discreate Cousine Transform)

eIQ: Uç Zeka (Edge Intelligence)

MFCC: Mel Frekansı Septral Katsayıları (Mel Frequency Cepstral Coefficients)

FFT: Hızlı Fourier Dönüşümü (Fast Fourier Transform)

FNN: İleri Sinir Ağları (Forward Neural Network)

LSTM: Uzun Kısa Süreli Hafıza (Long Short Term Memory)

RNN: Yinelemeli Sinir Ağları (Recurrent Neural Network)

ML: Makine Öğrenmesi (Machine Learning)

UL: Gözetimsiz Öğrenme (Unsupervised Learning)

SL: Gözetimli Öğrenme (Supervised Learning)

RL: Pekiştirmeli Öğrenme (Reinforcement Learning)

TF: TensorFlow

TFLite: TensorFlow Lite

RELU: Doğrultulmuş Doğrusal Birim (Rectified Linear Unit)

KWS: Anahtar Kelime Yakalama (Key Word Spotting)

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1: Derin Öğrenme Mimarileri…….**31**
Çizelge 3.2: RNN Mimarisi Avantaj ve Dezavantajları……..**48**
Çizelge 3.3: Yapay Zekâ Kütüphanleri……**45**
Çizelge 4.1: MIMXRT 1060 Eval Board Özellikleri……**62**
Çizelge 4.2: WM8960 Parametrik Özellikler……**68**
Çizelge 5.1: CNN Colab Eğitim Sonuçları……...**87**
Çizelge 5.2: LSTM Colab Eğitim Sonuçları……...**87**
Çizelge 5.3: CNN Colab Model Parametreleri……**90**
Çizelge 5.4: Gömülü Mimari RAM/ROM Optimizasyonu……..**90**
Çizelge 5.5: LSTM Colab Model Parametreleri……**92**
Çizelge 5.6: Veri Seti Kullanıcı Test Sonuçları……**93**

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : İnsanda Ses Oluşum Yapısı[7]……..……..... ……..…….**3**
Şekil 1.2 : İnsanlar Arası Konuşma Protokolü[7]…..…….…… ……...…….….**4**
Şekil 1.3 : Komut Kontrol Protokol Yapısı…….……...……..**7**
Şekil 1.4 : Genel Problemler[9]…….……..…..……**8**
Şekil 1.5 : Ses İzolasyonu Amacıyla Kullanılan Akustik Oda ...…….……**8**
Şekil 1.6 : Gürültü Bastırma Sistemlerinde Genel Konsept……...……..…..…..……**9**
Şekil 1.7 : Beamforming ile Konuşma Tanıma Sistemi ……..……..…..…..……**9**
Şekil 1.8 : Komut Kontrol Yapısı Flow Chart Yapısı……..…..…..……**10**
Şekil 2.1 : MFCC İşlem Adımları……...……..…..…..……**14**
Şekil 2.2 : Gürültünün Temiz Sinyalden Ayrılması…..……..…..……**14**
Şekil 2.3 : Frame Blocking (Çerçeveleme)…….……..…..…..……**16**
Şekil 2.4 : Hamming Window[13]…….…**17**
Şekil 2.5 : Pencereleme Metodu Uygulanmış Bir Sinyal……..…..…..……**18**
Şekil 2.6 : Mel Skalası……**20**
Şekil 2.7 : Colab Mel Filtre Bankası……..……...……..**21**
Şekil 2.8 : Mel Skala Aralıkları……...……**21**
Şekil 2.9 : DCT Matris Uygulaması[16]…….....**22**
Şekil 2.10 : MFCC…….…….....**22**
Şekil 2.11 : MFCC Spectogram Matrisi Katsayıları……...**23**
Şekil 2.12 : Google Colab…….…….....**24**
Şekil 2.13: Google Colab Ses Sinyali Grafiği…….....**24**
Şekil 2.14 :FIR Filtre ile Gürültü Optimizasyonu…….…….....**25**
Şekil 2.15 :Praat ile ses analizi…….…….…….....**26**
Şekil 2.16 :PratGiriş Menüsü……..…….…….....**26**
Şekil 2.17 : MCU Xpresso IDE…….…….....**27**
Şekil 3.1 : Yapay Zeka Ağaç Şematiği[19]…….……..…..... **30**
Şekil 3.2 : Supervised Learning Örnek Görsel…….……..**31**
Şekil 3.3 : Unsupervised Örnek Görsel..……..…..**32**
Şekil 3.4 : Reinforcement Learning Örnek Görsel…….……**33**

Şekil 3.5 : Sinir Ağlarına Genel Bakış [20]……..……..**34**
Şekil 3.6 : ANN Katman Yapısı [20]…….……**34**
Şekil 3.7 : Yapay Sinir Ağları Detaylı Mimarisi…….……....**35**
Şekil 3.8 : Birim Basamak Fonksiyonu[22]…….…….....**36**
Şekil 3.9 : Lineer Fonksiyonu[22]……..…….……....**37**
Şekil 3.10: ReLu Fonksiyonu [22]……..…..……**37**
Şekil 3.11: Sigmoid Fonksiyonu [22]……..…..……**38**
Şekil 3.12: Tanh Fonksiyonu [22]……..…..……**39**
Şekil 3.13: Gradient Descent 3 boyutlu……..…..……**41**
Şekil 3.14: Yapay Sinir Ağları Kullanılan Bir Sistemin Eğitiminin Sonlanması……..**41**
Şekil 3.15: FFNNS Mimarisi [25]…….……..**42**
Şekil 3.16: Örnek bir CNN Algoritması [26]…….……..**43**
Şekil 3.17: CNN Mimarisi [25] …….……..**43**
Şekil 3.18: Stride Boyutu ve Çıkış Matrisi…….……....**44**
Şekil 3.19: Pooling İşlemi…….. …….……..**45**
Şekil 3.20: RNN Mimarisi [25] …….……..**45**
Şekil 3.21: RNN Döngüsü [27] …….……..**46**
Şekil 3.22: Backpropagation İşlem Döngüsü… …….……..**47**
Şekil 3.23: RNN Döngüsü ve LSTM Kapıları[27]…….……..**49**
Şekil 3.24: Makine Öğrenmesi NXP eIQ…….……..**50**
Şekil 3.25: Keras Uygulama Metodları…….……**52**
Şekil 3.26: Keras Kütüphanesinin TensorFlow İmplementasyonu…….……..**52**
Şekil 3.27: Model Tasarım Aşamalar…….…….....**53**
Şekil 3.28: Dataset'e Bağlı Model Eğitim Çıktıları…….…….....**55**
Şekil 3.29: Tez Kapsamında Kullanılan Etiket Kelimler…….…….....**55**
Şekil 3.30: Dataset yaş ve cinsiyet dağılımı…….……**55**
Şekil 3.31: Keras Modelleri [52]…….……..**56**
Şekil 3.32: CNN Modelden Elde Edilmiş Başarım Oranı…….……..**58**
Şekil 3.33: Keras Modelinin Kayıt Edilmesi…….……..**59**
Şekil 3.34: Keras Modelinin Yüklenmesi…….……..**59**
Şekil 3.35: Tensorflow – TFLite Dönüşümü…….……..**60**
Şekil 3.36: Model Parametreleri Eleman Sayısı…….……..**60**
Şekil 3.37: Gömülü Sistem Konuşma …….……..**60**
Şekil 4.1: Gömülü Sistem Konuşma …….…….……**61**

Şekil 4.2 : Arm Mimari Yapısı…. ……62
Şekil 4.3: Cortex M7 Fonksiyon Diyagramı ……63
Şekil 4.4: Floating Point Gösterimi[39]……64
Şekil 4.5: MIMXRT 1060 EVAL Board HW Mimarisi……65
Şekil 4.6 : ECM Mikrofon İç Yapısı…..……66
Şekil 4.7 : Mikrofon Devresi……67
Şekil 4.8: EM6022P Mikrofonuna Ait Frekans – Response Grafiği[41]……..…..67
Şekil 4.9: Codec Devresi İç Blokları……68
Şekil 4.10: Codec Yapısı……69
Şekil 4.11: Ana İşlemci Yapısı……69
Şekil 4.12: Mikrofon Analog Data İşleme Blokları……70
Şekil 4.13: PGA Kazanç Devresi [42]……70
Şekil 4.14: ALC Operasyonları……72
Şekil 4.15: I2S Protokol Zamanlama Diyagramı……73
Şekil 4.16: I2S Receiver Diyagram……74
Şekil 4.17: I2S/ SAI Blok Diyagram……75
Şekil 4.18: SAI Saat Frekans Çalışma Yapısı……75
Şekil 4.19: I2S Haberleşme Modeli…. ……76
Şekil 4.20: Yazılım Akış Şeması……77
Şekil 4.21: i. MX RT 1060 Blok Diyagramı [43]……78
Şekil 4.22: QSPI Data Hattı [44]……79
Şekil 4.23: Model Çalışması……80
Şekil 4.24: Program Çıktıları……81
Şekil 5.1: Veri Seti Dağılımı……86
Şekil 5.2 : CNN Model Özeti ….. ……87
Şekil 5.3: LSTM Model Özeti……88
Şekil 5.4: CNN Netron ile Görselleştirme……91
Şekil 5.5: Basitleştirilmiş Çalışma Mekanizması……93

GÖMÜLÜ SİSTEMLERDE SESLİ KOMUT TANIMA

ÖZET

Kişilerarası iletişim en yaygın olarak konuşma ile sağlanır. Ses, konuşmanın temel ve çok önemli bileşenidir. Akciğerlerden gelen havayı ses organlarıyla şekillendirerek kulak veya hassas aletler tarafından algılanabilen titreşimler dönüştürülmesi sesin en basit tanımı olarak nitelendirilebilir.

Konuşma, boğaz ve ağızdaki bu titreşimlerin ve insan zihni tarafından belirli bir gramer altyapısında algılanabilecek karmaşık bir yapıdaki bir dönüşümdür. Yukarıdaki bilgiler ışığında, tezin amacı, insanların belirli komutları karakterize etmek için oluşturdukları ses verilerini analiz etmektir; bu analize uygun çıkacak özelliklere sahip gömülü işlemci mimarisini içeren bir sistem yetiştirmektir. Ses ve konuşma tanıma, günümüz dünyasında çok popüler hale gelen bir teknoloji olma yolunda ilerlemektedir. Özellikle mahkeme ve savcılık büroları gibi konuşmaların çok hızlı yazılması gereken platformlar veya kolay işlemlerin hızlı bir şekilde gerçekleştirilmesi gereken banka müşteri hizmetleri için telefon uygulamaları bu özelliğin en büyük müşterileridir. Tüm bunlara ek olarak, ses işleme teknolojisi hızla gömülü mimarilere de girmektedir. Örneğin; Televizyonun önünde otururken, uzaktan kumandayla bir program aramak yerine, aradığınız programı söyleyerek programı daha hızlı filtreleyebilir ve isteğinize çok hızlı bir şekilde ulaşabilirsiniz. Aynı şekilde, otomobil teknolojilerinde yüksek hızlarda sürerken, radyoyu sesinizle kontrol etmek güvenlik ve konfor açısından büyük bir gelişmedir. Tez kapsamında yapılan çalışmada, ses işleme teknolojisinin getirdiği bu konfor ve rahatlığı destekleyecek literatüre yeni çalışmaların eklenmesi hedeflenmektedir. Amacımız eğitilmiş sistem üzerinden ilgili komutları tespit etmek ve komuta karşılık gelen eylemi gerçekleştirmektir. Tezin en büyük başarısı, ses analizi ve bu analiz sonucunda oluşturulan veri seti ile ilgili komut kelimesinin ait olduğu dil ailesine bakılmaksızın istenen dilde komutların kullanılmasını sağlamasıdır. Bu çalışmanın en başarılı sonucunu vermek için ses analizi işlemleri; örneğin, sesin filtre seçimi, pencereleme fonksiyonları, özellik çıkarma fonksiyonları karşılaştırmalı olarak denenmiş ve tezde kullanılacak yönteme karar verilmiştir. Derin öğrenme, insan beyninin çalışmasını taklit eden yapay bir zeka işlevidir. Veri işleme ve karar verme için kullanılabilecek modeller oluşturmak üzere tasarlanmıştır. Çok büyük bir sinir ağı ve büyük miktarda erişilebilir veri gerektirir. Makine öğrenimi daha basit kavramlar kullanırken derin öğrenme, insanların nasıl düşündüklerini ve öğrendiklerini taklit etmek için tasarlanmış yapay sinir ağlarıyla çalışır. Sinir ağları, tıpkı insan beyninin nöronlardan oluştuğu gibi katmanlardan oluşur. Ayrı katmanlardaki düğümler bitişik katmanlara bağlanır. Ağın sahip olduğu katman sayısından daha derin olduğu söylenir. İnsan beynindeki tek bir nöron, diğer nöronlardan binlerce sinyal alır. Yapay bir sinir ağında, sinyaller düğümler arasında seyahat eder ve ilgili ağırlıkları atar. Daha ağır bir düğümün bir sonraki düğüm katmanı üzerinde daha fazla etkisi olacaktır. Son katman, bir çıktı üretmek için ağırlıklı girdileri derler. Derin öğrenme sistemleri, büyük miktarda veri işlendiğinden ve birkaç karmaşık matematiksel hesaplama içerdiğinden güçlü donanım gerektirir. Bununla

birlikte, bu tür gelişmiş ekipmanlarla bile, derin öğrenme eğitimi hesaplamaları haftalar sürebilir. Derin öğrenme sistemleri, doğru sonuçlar elde etmek için büyük miktarda veri gerektirir; Buna göre, bilgi büyük veri setleri olarak beslenir. Verileri işlerken, yapay sinir ağları, verileri oldukça karmaşık matematiksel hesaplamaları içeren bir dizi ikili doğru veya yanlış sorudan cevaplarla sınıflandırabilir. Örneğin, bir yüz tanıma programı yüzlerin kenarlarını ve çizgilerini, daha sonra yüzlerin daha önemli kısımlarını ve son olarak yüzlerin genel temsillerini tanımayı ve tanımayı öğrenerek çalışır. Zamanla, program kendini eğitir ve doğru cevap olasılığı artar. Bu durumda, yüz tanıma programı yüzleri zaman içinde doğru bir şekilde tanımlayacaktır.

Tezin gömülü mimari yönünden bu yana, bir mikrodenetleyici ve 32-bit 600 Mhz saat frekansına sahip çevresel elemanlar kullanılmaktadır. Google AI Lab tarafından sağlanan, masa ve mikrofona yakın veri kümesini gürültüsüz bir ortamda kullanarak; Komut olarak 5 İngilizce kelime seçildi ve MFCC (Mel Frequency Cepstral Coefficients) ve LSTM (Uzun süreli bellek ağları) olan model TensorFlow ve Keras kütüphaneleri kullanılarak eğitildi. Cortex mimarisi elde edildiğinde, Cortex M serisi ailesine ait M7 işlemci, düşük kesme gecikmesi, düşük maliyetli hata ayıklama özellikleri ve geriye dönük uyumluluk ile yüksek verimli, yüksek performanslı, gömülü bir işlemcidir.

Colab ağındaki eğitimin sonunda başarı oranı "%95.07" idi. Benzer şekilde MFCC (Mel Frequency Cepstral Coefficients) ve CNN (Evrişimli Sinir Ağları) kullanılarak başarı oranı '%88.03' olarak elde edildi. Bu modeller, TFLite özelliğini destekleyen eIQ platformuna sahip i.MX RT 1060 gömülü işlemci mimarisinde kayan nokta kuralları dikkate alınarak C / C ++ dilleri ile yazılıma dönüştürülmüştür. Sonuç Ram. Flash optimizasyonu, Model uygulanabilirliği ve uyumluluğu dikkate alınarak yapılan karşılaştırmanın bir sonucu olarak, her iki yöntemin de “MFCC ve CNN” yönteminin kullanıldığı sistemlerde ses tanıma projelerinde daha yararlı olduğu düşünülmüştür. Gömülü sistemler ve PC çalışmaları karşılaştırıldığında, performans kayıpları meydana geldi. Nedenleri üç ana kategoriye ayrılabilir: ses yalıtım sorunları, kullanıcı hataları ve çevresel etkiler. Tanıma ortamındaki gürültü, hem konuşmanın spektral özelliklerini deforme ederek hem de uç noktaların yanlış algılanmasına neden olarak tanıma performansını azaltır. Tanınan odanın akustik özellikleri konuşmanın spektral özelliklerini de etkiler. Bu problemlerin ışığında, gelecekteki çalışmalar daha geniş bir veri kümesi, konuşma yönü algılama, aktif gürültü filtreleme, ses çocuk kilidi, cihazın bulunduğu ortamın akustik özelliklerini öğrenme, konuşma komutlarını bluetooth mesh ağlarına bağlı cihazlar üzerinden aktarmaya odaklanacaktır.

VOICE COMMAND RECOGNATION IN EMBEDDED SYSTEMS

SUMMARY

Interpersonal communication is most commonly provided by speech. Sound is basic and crucial constituent of the speech. The simplest definition of sound which is the vibrations that can be perceived by the ear or sensitive instruments by shaping the air coming from the lungs with the organs of the sound.

Speech is a transformation of these vibrations in the throat, mouth and a complex structure that can be perceived by the human mind in a certain grammatical infrastructure. In the light of the above information, the purpose of the thesis is by analyzing the audio data created by people to characterize certain commands; It is to train a system that includes embedded processor architecture with features that will come out in line with this analysis.

Voice and speech recognition is on the way to becoming a technology that has become very popular in today's world. Phone applications especially for platforms such as andorid, ios, court and prosecution offices where conversations should be written very quickly, or bank customer service where easy operations should be handled quickly. In addition to all these, sound processing technology is rapidly entering embedded architectures. For example; While sitting in front of the TV, instead of looking for a program with the remote, you can filter the program faster by saying the program you are looking for and reach your request very quickly. Likewise, while driving at high speeds in automobile technologies, controlling the radio with your voice is a great improvement in terms of safety and comfort. In the study carried out within the scope of the thesis, it is aimed to add new studies to the literature that will support this comfort and convenience brought by the sound processing technology.

Our purpose is to detect the relevant commands through the trained system and to perform the action corresponding to the command. The biggest achievement of the thesis is that it enables the use of commands in the desired language regardless of the language family to which the related command word belongs, by means of the sound analysis and the data set created as a result of this analysis. In order to give the most successful result of this study, sound analysis operations; For example, filter selection, windowing functions, feature extraction functions of the sound have all been tried comparatively and the method to be used in the thesis has been decided. In addition to sound analysis, the artificial learning method used in architecture was also searched and compared in the related literature and resources, and used in the thesis.

Deep learning is an artificial intelligence function that mimics the work of the human brain. It is designed to create models that can be used for data processing and decision making. It requires a very large neural network and a large amount of accessible data. While machine learning uses simpler concepts, deep learning works with artificial neural networks designed to mimic how people think and learn. Neural networks consist of layers, just as the human brain is made up of neurons. Nodes in individual layers are connected to adjacent layers. The network is said to be deeper than the

number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons. In an artificial neural network, the signals travel between the nodes and assign the respective weights. A heavier weight node will have more impact on the next node layer. The last layer compiles weighted inputs to produce an output. Deep learning systems require powerful hardware because a large amount of data is processed and contains several complex mathematical calculations. However, even with such advanced equipment, deep learning training calculations can take weeks.

Deep learning systems require a large amount of data to get accurate results; Accordingly, information is fed as large data sets. When processing data, artificial neural networks can classify data with answers from a series of binary true or false questions that involve highly complex mathematical calculations. For example, a face recognition program works by learning to identify and recognize the edges and lines of faces, then the more important parts of faces, and finally the general representations of faces. Over time, the program trains itself and the probability of correct answers increases. In this case, the face recognition program will correctly identify faces over time.

Since the embedded architectural side of the thesis, a microcontroller and peripheral elements with a 32-bit 600 Mhz clock frequency are used. By using dataset, which is provided by Google AI Lab, close to the desk and microphone, in a noise-free environment; 5 English words were chosen as commands and the model with MFCC (Mel Frequency Cepstral Coefficients) and LSTM (Long-term memory networks) were trained using TensorFlow and Keras libraries.

When obtain the Cortex architecture, the M7 processor, which belongs to the Cortex M series family, is a high-efficiency, high-performance, embedded processor with low cutting latency, low-cost debugging features and backward compatibility.

At the end of the training on the Colab network, the success rate was "95.07%". Similarly, using the MFCC (Mel Frequency Cepstral Coefficients) and CNN (Convolutional Neural Networks), the success rate was achieved as '88.03%'. These models have been converted into software with C / C ++ languages by considering floating point rules on the i.MX RT 1060 embedded processor architecture with eIQ platform supporting TFLite feature.

Keras ensures that each training phase is recorded during a deep learning model training. We can achieve the default training steps recorded while training all the deep learning models. Records training metrics for each training phase. This includes the loss and accuracy of the verification data set (for classification problems), if set. The history object is returned from the calls to the fit () function, which is used to train the model. Metrics are stored in a dictionary on the past member of the returned object. While training the model as keras, the accuracy and loss in the keras model for verification data may vary in different cases. Generally, as each age increases, losses should decrease and accuracy should increase. When training a machine learning model, one of the main things to avoid is to adapt to extreme. This is when the model fits well to train data, but cannot generalize and make accurate predictions for data that it has not seen before. To find out if their models fit more than necessary, data scientists use a technique called cross-validation, where they divide their data into two parts - the training set and the validation set. The training set is used to train the model, while the validation set is used only to evaluate the performance of the model.

The metrics in the training set allow you to see how your model is progressing in terms of education, but metrics in the validation set that allow you to measure the quality of your model - make new predictions based on data. At this point, as can be seen, the terms loss and accuracy are the measures of loss and accuracy in the training set, while test_ loss (val_loss) and test_accuracy (val_acc) are the loss and accuracy measurements in the verification set.

As a result of the comparison made by considering both Ram and Flash optimization, Model applicability and compatibility, both methods have been considered to be more useful in voice recognition projects within the systems embedded in the use of "MFCC and CNN" method. When the embedded systems and PC studies were compared, performance losses occurred. The causes can be grouped into three main categories: sound insulation problems, user errors, and environmental impacts. Noise in the recognition environment reduces recognition performance, both by deforming the spectral properties of speech and by causing false detection of endpoints. The acoustic properties of the recognized room also affect the spectral properties of speech. In the light of these problems, future studies will focus on a broader dataset, speech direction detection, active noise filtering, voice childlock, learning the acoustic properties of the environment where the device is located, transferring speech commands over devices connected to bluetooth mesh networks.

1 GİRİŞ

Konuşmacı tarafından söylenen sözleri tanımak için işitsel sinyallerin analizine Konuşma Tanıma denir.Yani en basit haliyle beyinde oluşturulan sinyallerin ağız, gırtlak, akciğerler ve ses tellerinin yardımıyla oluşturulan akustik dalgaların insan beyninde bir kavramı nitelendirmesidir. Tezin amacı insanlar arasında yüzyıllar boyunca evrimleşmenin sonucu olarak ortaya çıkmış bu yapıyı anlamak ve insan makine arasında aynı protokolü başarılı bir şekilde oluşturabilmektir.

1.1 Literatür Karşılaştırması

Ses ve konuşma tanımaları günümüz dünyasında iyice popülerleşen bir teknoloji olma yolunda ilerlemektedir. Özellikle andorid, ios gibi platformlar için telefon uygulamaları, çok hızlı bir şekilde konuşmaların yazıya dökülmesi gereken mahkeme ve savcılık ofisleri veya kolay işlemlerin hızlıca halledilmesi gerektiği banka müşteri hizmetleri gibi. Bütün bu sayılanlara ek olarak ses işleme teknolojisi, gömülü mimarilere de hızla girmektedir. Örnek vermek gerekirse; tv karşısında otururken kumanda ile program aramak yerine, aradığınız programı sesli olarak söylerek çok daha hızlı filtre yapabilir ve isteğinize çok hızlı ulaşabilirsiniz. Aynı şekilde otomobil teknolojilerinde de yüksek hızlarda giderken radyo kontrolünü sesiniz ile yapmak güvenlik ve konfor açısından çok büyük bir gelişmedir. Tez kapsamında yapılan çalışmada ses işleme teknolojisinin getirdiği bu konfor ve kolaylığa destek verecek yeni çalışmaları literatüre kazandırmaktır. Bu kapsamda aşağıda yapılmış benzer 5 çalışma ele alınmış ve bu çalışmalara göre elde edilmeye çalışılan kazanımlar verilmiştir. İlk olarak ele alınan çalışma İstanbul Teknik Üniversitesi yüksek lisans tezi kapsamında yazılmış “EMBEDDED SPEECH RECOGNITION” başlıklı tezdir. Bu tez, çamaşır makineleri için offline bir ARM tabanlı geliştirme kartı üzerine sesli komut tanıma üzerine yazılımmış tezdir [1].

Bu teze göre getirilmiş yenilikler; wakeup word kelime teknolojisi ile sistemin ilgili uyandırma kelimesi gelene kadar uykuda kalması ve enerji verimliliği, Tensorflowlite ve Colab platformları kullanılarak tasarlanan AI modelin daha kompakt ve uygulanabilir olması, buna ek olarak kolay bir şekilde görselleştirilerek PC ve gömülü işlemci mimarisi arasındaki ortam farkının çok daha iyi anlaşılabilmesini sağlamıştır. Ayrıca iki ayrı AI yapısının paralel çalıştırması geliştirdiğimiz tez kapsamında çok önemli bir ayrıcalıktır.

İkinci olarak ele alınan çalışma ise "Microcontroller Implementation of a Voice Command Recognition System for Human-Machine Interface in Embedded Systems" adlı yapılan çalışmadır.Bu çalışma optimum ram rom ihtiyacını göz önüne alarak AI tabanlı offline gömülü sistem tasarımı üzerinedir.[2] Bu çalışmaya göre tezde ele aldığımız mimari, wakeword detection ve Paralel AI yapıları sayesinde çok daha az enerji tüketimi ve yüksek güvenlik sağlanmaktadır. Bunlara ek olarak geliştirilen deep learning algoritması TFLite ile gömülü mimariye aktarılmış bu da RAM ve Flash optimizasyonunda ciddi bir katkı sağlamıştır.

Üçüncü olarak ele alınan çalışma ise "Smartphone-Based Online and Offline Speech Recognition System for ROS-Based Robots" adlı yapılan çalışmadır. Bu çalışma kapsamında bir cep telefonu vasıtasıyla verilen tanımlı komutlar ile robot araç ilgili komutu yerine getirmektedir. [3] Bu çalışma da var olan yapıya ek olarak tezde geliştirdiğimiz yenilikler hazır bir sinyal processing aplikasyonu olmadan floating point algoritması göz önüne alınarak sinyal işleme kütüphanesi tasarlanmasıdır. Ayrıca konuşmanın etiketlenmesi için herhangi bir server kullanılmadan far field (1.5 metreye kadar) tasarlanan sistem kullanıcıya büyük kolaylık sağlamaktadır.

Dördüncü olarak ele alınan çalışma ise "Hardware Speech Recognition for User Interfaces in Low Cost, Low Power Devices" çalışması dil modeli tabanlı bir algortima sunmaktadır.[4] Bu çalışmada var olan yapının aksine tezde tasarlanan mimari konuşma veri seti yapısıdır. Yeterli miktarda ses örneğinin toplanması durumunda hiçbir dilin gramer yapısına bağlı kalmaksızın sonuç alınmaktadır.

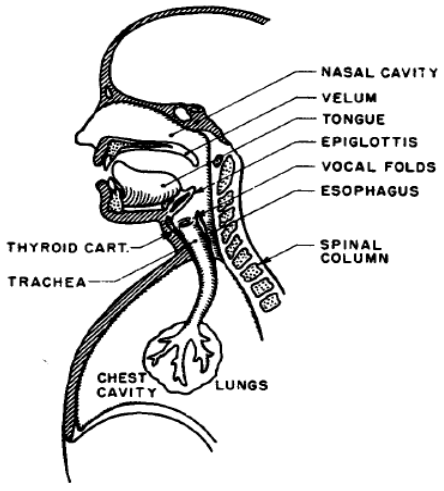
Beşinci ve son ele alınan çalışma ise "A ROBUST SPEECH COMMAND RECOGNIZER FOR EMBEDDED APPLICATIONS" çalışmasında belirli Portekizce kelimelerin anlaşılması için başarılı bir sistem geliştirlmiştir.[5] Bu çalışmada da sistem dilin gramer yapısına tabii tutulmuştur.

Tez kapsamında gerçekleştirilen çalışma ise tamamen dil bilgisi kurallarından bağımsız şekilde uygun veri setinin ve yapay öğrenme modelinin seçimi en başarılı algoritmayı yakalamayı amaçlamıştır. Yukarıda ilham alınan beş değerli akademik çalışmaya yer verilmiş ve tez aşamasında bu çalışmaların üzerine geliştirilen önemli farklara değinilmiştir.

1.2 Konuşma Tanımada Temel Kavramlar

İnsanlar arasında geliştirilen sesli iletişim modelini anlamak ve temel kavramlara hakim olmak insan- makine arası iletişim modelini kurmak açısından büyük önem taşımaktadır. Bu kavramlar insan sesi, fonem, ses tanıma, ses analizi, ses kodlama ve ses sentezi olarak sıralanabilir.

İnsan Sesi: Bütün omurgalılar ağız, akciğerler ve ses tellerini kullanarak ses çıkarırlar. Sesin oluşması için önce akciğerlerden gelen hava soluk borusuna dolar ve buradan dışarı çıkar. Soluk borusunun üst bölümünde ise gırtlak yer alır. Gırtlakta ses telleri yer alır. Bu teller akciğerlerden gelen hava ile titreşir ve insan sesinin çıkmasını sağlar. Sesin kaynağı spektral düzlemde temel ve harmonik bileşenlerden oluşur ve gırtlaktan başlayan hareketli ve hareketsiz organların meydana getirdiği konuşma boşluğunun şekline göre ses tonlarını oluştururlar. [6] Gırtlak kası, dudak ve ağız yardımıyla bu sesler sözcüklere dönüştürülür. Şekil 1.1'de bu yapı görülebilmektedir.



Şekil 1.1 : İnsanda Ses Oluşum Yapısı [7].

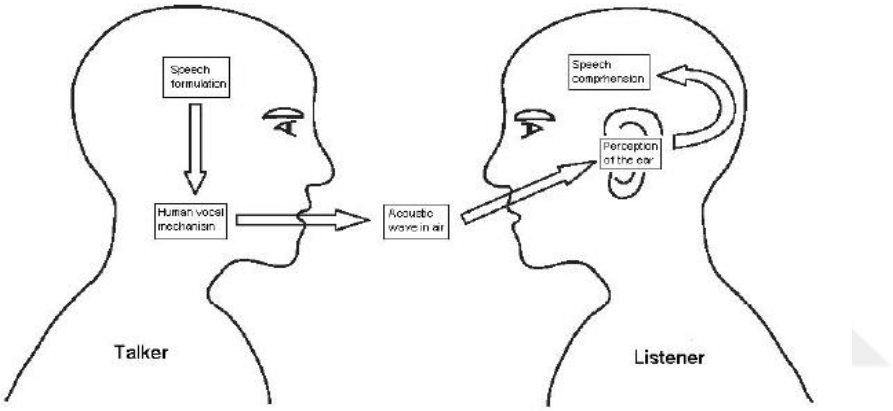
Fonem: Anlam içeren ve değişmesi ile dildeki bulunduğu bir kelimenin anlamını değiştiren en küçük ses birimine denir.

Ses Tanıma: Mikrofon veya benzeri bir ses kayıt cihazı tarafından alınmış akustik bir sinyalin kelime topluluğuna çevrim işlemi olarak nitelendirilebilir.

Ses Analizi: Sesin özelliklerinin çıkarılabilmesi için yapılan işlemlerdir.

Ses Kodlama: İşlem sırasında gözlenebilir bir sinyal kalitesi kaybı olmadan, sinyalin sayısal gösterimi olan bit oranın azaltmayı amaç edinen kodlama algoritmalarına denir.

Ses Sentezi: Metinden yapay olarak ses üretme işlemine verilen isimdir. Şekil 1.2'de insanlar arası protokol yapısı detaylı bir şekilde görülmektedir.



Şekil 1.2 : İnsanlar Arası Konuşma Protokolü [7].

1.3 Konuşma Tanıma Kullanım Alanları

Konuşma tanıma kullanım uygulamaları çok geniş bir çerçevede kullanılmaktadır. Kullanıldığı alanlarda hız, uygulama kolaylığı, bekleme süresini azaltma gibi katkılar sunmaktadır. Bu alanlar dikte-yazdırma, komut-kontrol, telefonla hizmet, tıbbi alanlar ve gömülü sistemlerdir. [8]

Dikte – Yazdırma: Toplantı, mahkeme gibi organizasyonlarda konuşmaları algılayıp yazıya geçirme işlemini yapan uygulamalardır.

(Ticari Uygulamalar: Microsof Dictation, IBMviaVoice vb.)

Komut- Kontrol: Bunlar, bir komut kümesi ve komuta karşılık gelen bir dizi işlev tarafından oluşturulan uygulamalardır. Komutta mikrofon tarafından alınan ses ifadesi verileri algılanır ve algılanan komuta karşılık gelen işlev yürütülür.

Telefonla Hizmet: Özellikle banka ve devlet kuruluşları olmak üzere müşteri temsilcisine bağlanmadan yapılabilecek işlemler için geliştirilmiş teknolojidir.

Bu uygulamada sizi bir dijital asistan karşılar. Size işlem yapmak istediğiniz anahtar kelimeleri sorar ve sizin verdiğiniz cevap doğrultusunda sorununuzu çözmeye çalışır.

Tıbbi Alanlar: Tıbbi yetersizliklerden dolayı ellerini ve klasik veri girişi elamanlarından olan klavye, fare vb. operatörleri kullanmayan insanlara, konuşma tanıma vasıtasıyla veri girişi imkânı sunulmaktadır. Özel donanımlar ve kurulabilecek sistemler sayesinde, bu durumda olan bir kişi sesli komutlarla, örnek olarak televizyonu kontrol edebilir, müzik setini kumada edebilir vb. işlemleri kolaylıkla yapabilir.

1.4 Konuşma Tanıma Sistemlerinin Sınıflandırılması

Konuşma tanıma sistemlerini 4 alt başlıkta incelemekteyiz. [8]

a) Sesin Sürekliliğine Göre

b) Konuşmacıya Bağımlılığına Göre

c) Ses Tanıma Sisteminde Temel Alınan Birime Göre

d) Metin Bağımlılığına Göre

1.4.1 Sesin sürekliliğine göre

Sesin sürekliliğine göre 4 alt başlık altında incelenmektedir.

1.4.1.1 Ayrık kelimeler

Bu tür konuşma tanıma sistemlerinde, sistem tarafından kelimeler arası kısa boşluklar beklenir.

1.4.1.2 Sürekli konuşma

Konuşmacının en doğal haliyle konuşması beklenir. Dikte – yazdırma uygulamaları bu sınıflandırmaya girer.

1.4.1.3 Bağlı kelimeler

Ayrık kelimelere çok benzer. Ayrık kelimelere göre kelimeler arası boşluklar daha kısadır. Konuşmacının daha kısa boşluklarla kelimeleri seslendirmesini destekleyen sistemler bu sınıfa girer.

1.4.1.4 Doğal konuşma

Sürekli konuşma sınıflandırmasına çok benzer. Fakat insan doğasından kaynaklanan, doğal konuşma özelliklerini de algılayabilecek seviyede geliştirilmişlerdir. Bu tür konuşma özelliklerine, 'hımm', 'eee' vb. mırıldanmalar dahildir.

1.4.2 Konuşmacıya bağımlılığına göre

Bu sınıfta kişiye bağımlı ve kişiden bağımsız olarak 2 alt sınıf oluşmaktadır.

1.4.2.1 Kişiye bağımlı

Bu tür ses tanıma sistemlerinde farklı bir konuşmacının sesi tanınmaz. Sadece sistemde kayıtlı kişiler için sistem tepki verir.

1.4.2.2 Kişiden bağımsız

Farklı bir konuşmacı için şablonların güncellenmesi gerekmez. Sistem yaş, cinsiyet, aksan ayırmadan tepki verir. Kişiye bağımlı sistemlere göre çok daha zor bir geliştirme süreci vardır. Ses sinyalinin özelliği kişiden kişiye hatta aynı kişinin farkli zaman, duygu ve düşünce hallerinde bile farklılık gösterebilir.

1.4.3 Temel alınan birime göre

Aşağıda yer alan alt başlıklarda detaylı anlatılmaktadır.

1.4.3.1 Sözcük tabanlı

Bu sistemlerde en küçük birim sözcüklerdir. Oluşturulan referans şablonlar sözcüktür.

Doğruluk dereceleri daha yüksektir, fakat bu sistem gereksinimini arttırır.

Genellikle Komut – Kontrol uygulamalarında kullanılırlar.

1.4.3.2 Fonem tabanlı

Bu sistemlerde en küçük birim ise fonemlerdir. Fonem, bir dilde anlam ayıracı en küçük ses birimine denir. Bundan dolayı da şablon sayısı, oluşturulması, saklanması, işlenmesi çok az bir sistem gereksinimi ile karşılanır.

Fonemlerin tabanlı yapıların kullanılmasında ki en büyük problem ise başlangıç ve bitişlerinin tespitinin zor olmasıdır. Ayrıca doğruluk oranı sözcük tabanlı sistemlere göre düşüktür. Olumlu özelliği ise hata durumlarında geri dönüşler yaparak düzeltme imkanı olmasıdır.

1.4.4 Metin bağlılığına göre

Aşağıda yer alan alt başlıklarda detaylı anlatılmaktadır.

1.4.4.1 Metne dayalı tanıma

Bu tür sistemlerde kullanılan test verisi, eğtim verisi ile sınırlı tıtulur. Yani sistem, eğitimde kullanılan kelimelerin farklı seslendirilişleri ile test edilir.

1.4.4.2 Metinden bağımsız tanıma

Ağın eğitimde kullanılan kelimelerin yanı sıra bu kelimelerin eğitim dışı farklı kombinasyonlarına da cevap verebilir.

Örnek vermek gerekirse;

Sistem “beş” ve “yirmi” kelimeleri ile eğitilmişse bu sistem “yirmibeş” kelimesini de tanır.

1.5 Kurulacak Modele Karar Verilmesi

Tezin amacı gömülü bir işlemci mimarisine sahip sistemde komut – kontrol yöntemi ile kullanıcının komutunu tanımak ve ilgili sistemde gerekli aksiyonun yapılmasıdır. Bu hedef ve yukarıdaki bilgiler doğrultusunda aşağıdaki **Şekil 1.3’de** gösterilen sınıflandırmaya karar verilmiştir.

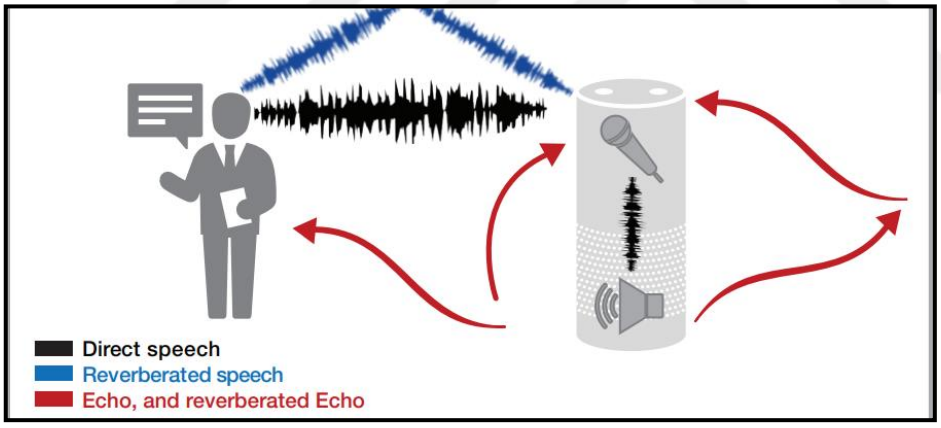
Sınıflandırma Çizelgesi			
Konuşmacıya Bağımlılık	Bağımlı	Bağımsız	
Temel Alınan Ses Birimi	Sözcük Tabanlı	Fonem Tabanlı	
Sesin Sürekliliğine Göre	Bağlı Kelime Tanıma	Ayrık Kelime Tanıma	Sürekli Konuşma Tanıma
Metin Bağlılığına Göre	Metne Dayalı	Metinden Bağımsız	

Şekil 1.3 : Komut Kontrol Protokol Yapısı.

1.6 Konuşma Tanıma Sistemlerinde Temel Sorunlar

Ses tanıma sistemlerinin geliştirilmesi aşamasında yaşanan temel bazı sorunlar mevcuttur. Bu sorunlardan en önemlisi ise ses sinyalinin ortam gürültüsünden çok fazla etkilenmesidir. Bu problem kapsamında gürültünün temizlenmesi için ses sinyalinin bir filtreden geçirilmesi gerekmektedir. Diğer bir problem ise ses sinyalinin kişiden kişiye hatta aynı kişinin farklı duygu durum hallerinde bile çok fazla farklılık göstermesidir. Aynı zamanda şive ve aksanda bu farklılıkları arttıran çok büyük etmenlerdir. Çoğu gömülü uygulama, ideal olmayan akustik ortamlarda çalışır.

Doğal olarak yukarıda sayılan engellere ek olarak yetersiz flash ve ram, işlemci hızı, makine gürültüsü, kullanılacak olan mikrofonların yetersizliği gibi durumlarla karşı karşıya kalınır. Ortam gürültüsü çoğu zaman konuşma sinyalinin kalitesini, çok ciddi şekilde bozabilir. Gömülü uygulamalarda görülen problemler genel olarak yankı engelleme (Echo Cancellation), gürültü bastırma (Noise Suppression), hüzme şekillendirme (Beamforming) başlıkları altında kısaca özetlenmiştir. Şekil 1.4'de ses işleme kapsamında yaşanan problemler özetlenmiştir.

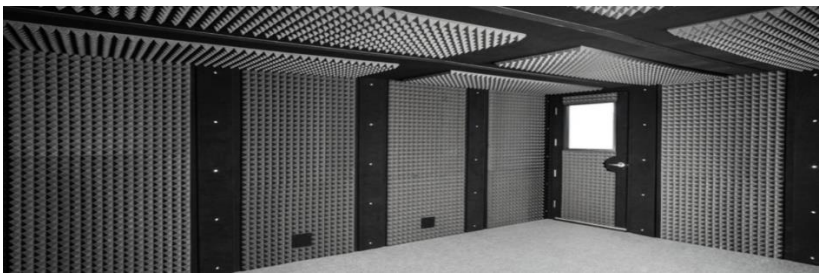


Şekil 1.4 : Genel Problemler [9].

1.7 Eko Engelleme (Echo Cancellation)

Eko, çeşitli elektriksel ve akustik nedenlerden dolayı oluşan yankıdır. Örnek vermek gerekirse birçok gömülü konuşma uygulamalarında mikrofon ve hoparlör beraber bulunmaktadır. Bu durum mikrofondan alınan input datasına ek olarak hoparlörden yansıyan ses sinyalleri de giriş verisi olarak alınır ve gönderilmesi gereken asıl data bozulmuş olur. Alınması gereken verinin üzerine binen yansıma sinyalleri yankıya sebep olurlar. Bu duruma ek olarak konuşmacının bulunduğu ortam ses izolasyonu

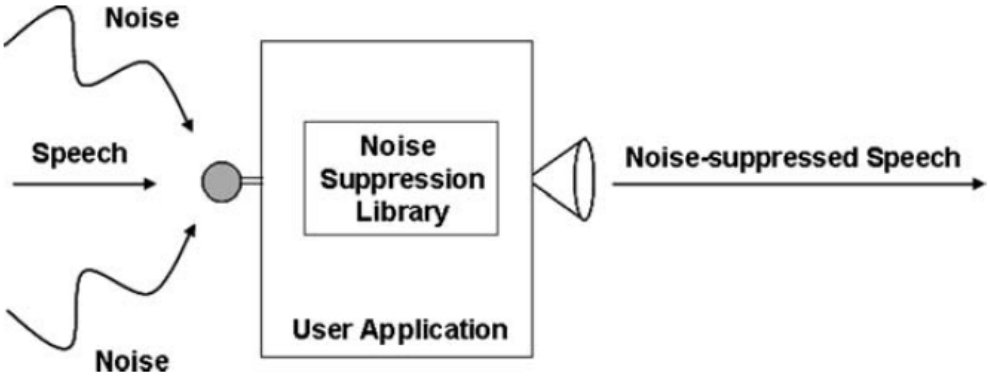
açısından uygun olmayabilir. Böylece ortam da yankıya neden olur ve bu da gönderilecek olan giriş verisini bozabilir. Şekil 1.5'de akustik oda tasarımı verilmiştir.



Şekil 1.5 : Ses izolasyonu amacıyla kullanılan akustik oda.

1.8 Gürültü Bastırma (Noise Suppression)

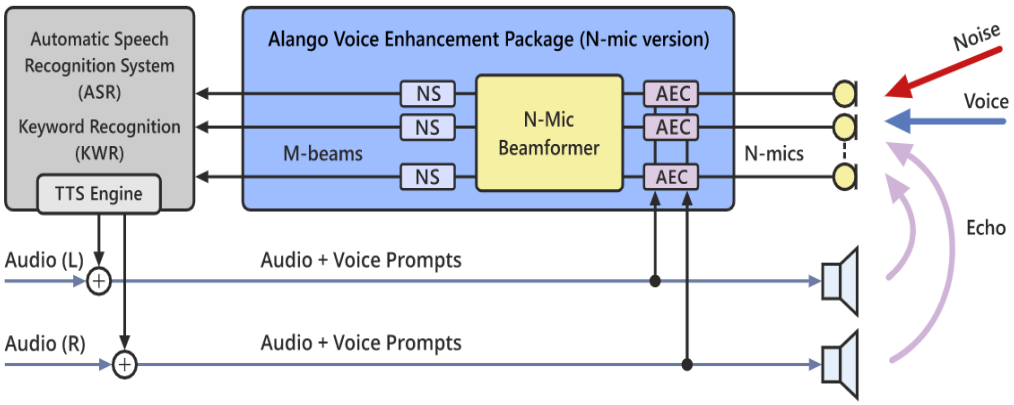
Gürültü bastırma, alınmak istenen bir konuşma sinyalini bozabilecek istenmeyen sinyalleri azaltmak için kullanılan filtre tasarımıdır. Genellikle tipik bir konuşma sinyali iki ana bölümden oluşur. Bir konuşma bilgilerini taşır ve diğeri sözel bilgi olmadan sözler arasında sessiz veya gürültülü bölümleri içerir. Şekil 1.6'da verilen sistem bu yapıyı özetlemektedir.



Şekil 1.6 : Gürültü Bastırma Sistemlerinde Genel Konsept.

1.8.1 Hüzme biçimlendirme (Beamforming)

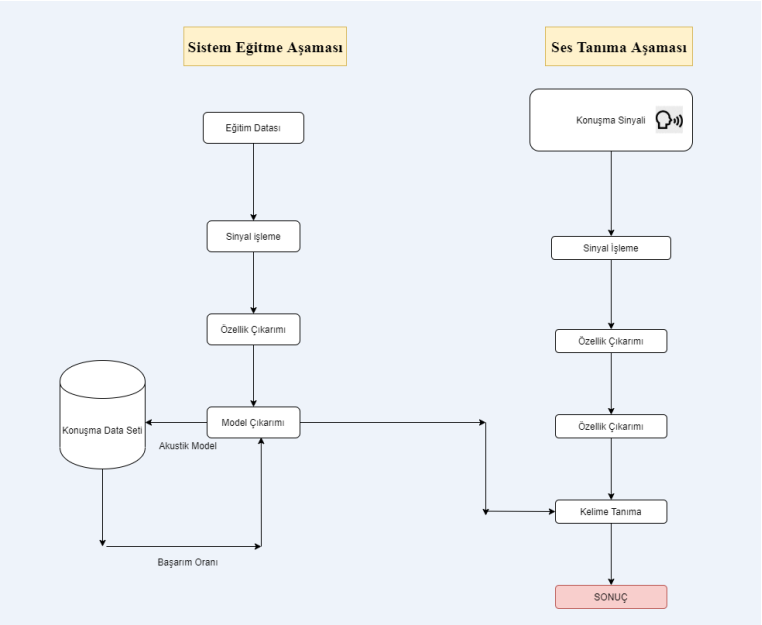
Hüzme Biçimlendirme (Beamforming) bir sinyal işleme yöntemidir. Bir sinyalin varlığını tespit etmek, geliş yönünü bulmak, gürültü ve yankılamayla sinyalin bozulmasını engellemek amacıyla kullanılan bir tekniktir. Bu teknik, temel olarak soruna mekânsal bakış açısıyla yaklaşır. Bir mikrofon dizisi, belirli bir yönden gelen giriş sinyalini belirler ve diğer yönlerden gelen gürültü yaratıcı sinyalleri sönümler. Şekil 1.7'de bu yapı detaylıca görülebilir.



Şekil 1.7 : Beamforming ile Konuşma Tanıma Sistemi.

1.9 Tez Organizasyonu

Tez organizasyonu gömülü işlemci mimarisi kullanılarak %95 başarım oranı başarım sağlayacak sesle komut-kontrol yapısı geliştirmeye yönelik tasarlanmıştır. Birinci bölüm olan "Giriş" bölümünde konuşmanın tarihçeşi, nasıl sınıflandırıldığı, kullanım alanları ve bu tip sistemlerde karşılaşılan temel sorunlara değinilmiştir. İkinci bölüm olan "Sinyal İşleme" başlığı altında ise analog – dijital sinyal, genlik, dalgaboyu, frekans vb. kavramlara değinilmiştir. Buna ek olarak sinyal işleme de kullanılan teknikler detaylıca anlatılmıştır. Kullanılan programlar ve çalışmalar detaylıca anlatılmaya çalışılmıştır. Üçüncü bölüm olan "Derin Öğrenme" başlığı altında makine öğrenmesi tarihçesi, alt başlıkları, derin öğrenmeye geçiş, kullanılan metdolar ve programlar, yapılan çalışmaların sonuçları detaylı bir şekilde verilmeye çalışılmıştır. Dördüncü bölüm olan "Gömülü Sistemler Mimarisi Tasarımı" başlığı altında ise kullanılan işlemci mimarisi, Gömülü sistemlerde audio processing mimarisi, floating point yapısı, kullanılan ide vb. bilgiler detaylıca verilmiştir. Beşinci bölüm olan "Deneysel Sonuçlar ve Tartışma" bölümünde ise elde edilen sonuçlar detaylıca karşılaştırmalı olarak verilmeye çalışılmıştır. Son bölüm olan altıncı bölüm "Sonuç ve Öneriler" başlığı altında ise elde edilen bütün çıktılar ve öğrenilen yeni kavramlar açıklanmıştır. Şekil 1.8'de ilgili algoritma yapısı görülmektedir.



Şekil 1.8 : Komut- Kontrol Mekanizması Flow - Chart Yapısı.

2. SİNYAL İŞLEME

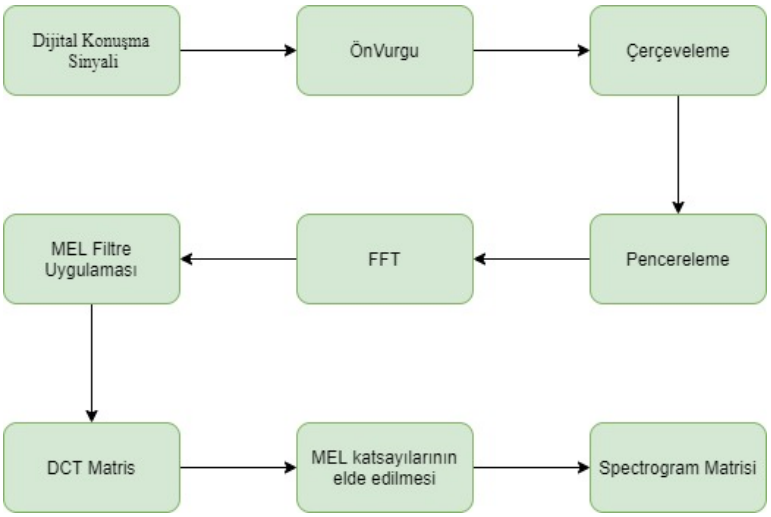
Konuşma işleme kavramlarını anlamak genel sinyal işleme temel ilkelerine hakim olmaktan geçmektedir. Bu kapsamda MFCC öznitelik vektörünün en iyi şekilde çıkarılması ve spektogram matrisinin elde edilmesi aşamasına kadar olan kısımlar ele alınacaktır.

2.1 Sayısal Sinyallerin İşlenmesi

Sayısal sinyal sahip olduğu özelliklerin çıkarımı için bazı işlemlerden geçirilir. Özellik çıkarma, ilk ham veri kümesinin işlenmek üzere daha yönetilebilir gruplara indirgeneceği boyutsallık azaltma sürecine verilen addır. Bu büyük veri kümelerinin bir özelliği, işlemek için çok fazla bilgi işlem kaynağı gerektiren çok sayıda değişkendir.[11] Özellik çıkarma, değişkenleri seçen veya özelliklerle birleştiren yöntemlerin genel adıdır ve işlenmesi gereken veri miktarını etkili bir şekilde azaltırken, orijinal veri kümesini doğru ve eksiksiz bir şekilde açıklar. Özetlemek gerekirse Elimizdeki sayısal datayı işimize yarayacak en optimum ve küçük boyutlara indirgemeye çalıştığımız kısımdır. Bu başlık altında ele alacağımız kısımlar ses sinyalinden akustik özellik çıkarımı üzerine olacaktır. Bu akustik öznitelikler, konuşma sinyaline ait özgün bilgiler taşımaktadır ve oluşturulacak olan akustik modelde giriş olarak kullanılacaktır. Çalışmada MFCC öznitelik çıkarma yöntemi kullanılacaktır. [12]

2.1.1 MFCC öznitelik çıkarma yöntemi

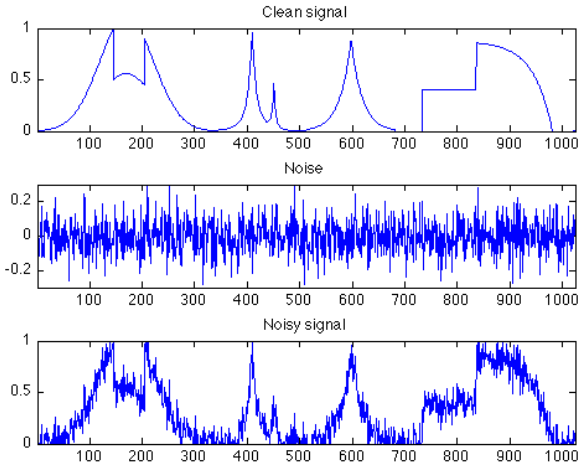
MFCC (Mel Frequency Cepstral Coefficients) öznitelik çıkarma yöntemi konuşma tanıma sisteminin başarısında önemli bir etkendir. Öznitelik çıkarma adımları Şekil 2.1’ de verilmiştir.



Şekil 2.1 : MFCC işlem adımları.

Önvurgu (Preemphasis): Sinyal işleme konusunda en önemli sorunlardan bir tanesi sinyali gürültüden temizlemektir. İletilmek istenen bilgi sinyali ile bu sinyalin üzerindeki gürültü bileşenlerinin genlikleri arasındaki orana sinyal gürültü oranı (S/N) denir. Ön vurgu adımındaki en büyük amaç bu bu oranı mümkün olduğu kadar yüksek tutmaktır. Bu kapsamda çözüm ise, yüksek frekanslı ancak düşük genlikli harmoniklerin, daha sonra kolay tanınıp temizlenmesi için abartılarak yükseltilmesidir. Önvurgu tekniği sinyal işlemede filtre kullanımı ile sağlanmaktadır.

Filtre Tasarımı: Bir filtre zaman alanında ve frekans alanında olmak üzere 2 yol ile tasarlanabilir. IIR filtreler için genlik ve fazı frekans alanında tanımlanırken, FIR filtrelerin impulse cevabı da zaman alanında tanımlanarak tasarım yapılır. Şekil 2.2'de verilen yapıda gürültünün sinyalden ayrılması detaylı bir şekilde gösterilmiştir.



Şekil 2.2 : Gürültünün temiz sinyalden ayrılması.

IIR Filtre Tasarımı: Infinite impulse response filtreler yani sınırsız sinyal tepkisi yaratan filtreler, akım çıkışı örnek değerini elde etmek için akım giriş örneği değerini, geçmiş giriş ve çıkış örneklerini kullanır Bir başka deyişle, çıktı verileri özyineleme ışığında girdide tekrar kullanılarak sürekli, dinamik bir akış sağlanır. Sistemin son tepkisi girdiler yanı sınra önceki çıktılar ile belirlenir. IIR filtreleri ayrık olarak tasarlamanın 2 yolu vardır.

1- Filtre analog olarak tasarlanıp örnekleme ya da dönüşüm yöntemleriyle ayrıklaştırılır.

2- Filtre, doğrudan yaklaşım ya da eniyileme yöntemleriyle ayrık olarak tasarlanır.

En çok kullanılan analog filtre tipleri Buttterworth, Chebyshev ve Elliptic filtrelerdir. Alçak geçiren genlik tanımlamarı en düşük dereceli olarak ilk Elliptic filtreyle sonra sırasıyla Chebyshev ve Butterworth filtre ile gerçeklenebilir. Ancak aralarında en düz genlik cevabına sahip filtre Butterwort filtredir.

FIR Filtre Tasarımı: Finite impulse response yani sınırlı sinyal tepkisi yaratan filtreler, bir akım çıkış örneği değeri elde etmek için sadece mevcut ve geçmiş giriş dijital örneklerini kullanır. Geçmiş çıktı örneklerini kullanmaz. FIR filtre özellikleri aşağıda verildiği gibidir.

1- FIR filtre tasarımı ayrık bir işlemdir.

2- Analog bölgede eşdeğer bir FIR filtre karşılığı yoktur.

3- FIR filtre tanımlamaları daha çok zaman bölgede yapılır.

4- FIR filtrelerin iki avantajı kararlılıkları ve lineer fazlı olmalarıdır. Aynı özellikleri sağlayan IIR filtrelere kıyasla FIR filtrelerin katsayı sayısı (dereceleri) çok daha fazladır.

5- En basit FIR filtre Moving Average (MA) filtresidir:

$$h(n) = 1 / M \quad 0 <= n <= M - 1$$

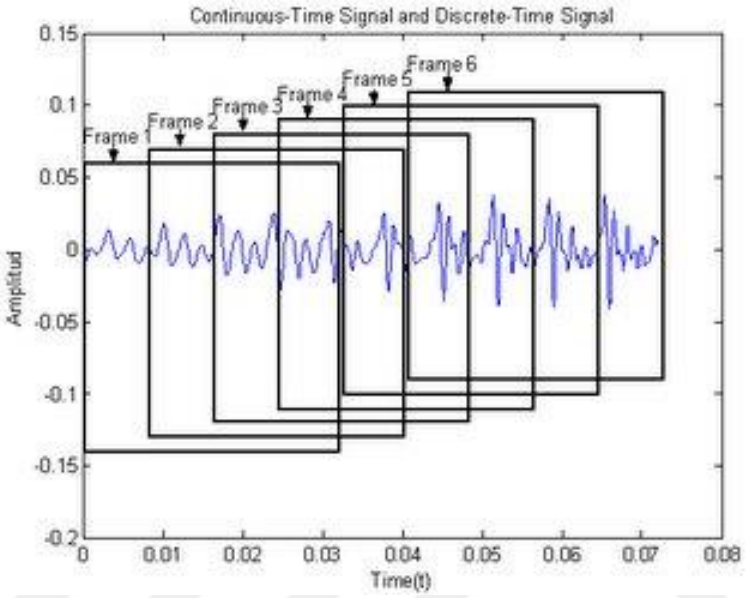
$$H(z)=\sum_{n=0}^{M-1} \frac{1}{M} * z^{-n} \tag{2.1}$$

6- Genlik cevabı anlamında oldukça zayıf olan bir alçak geçiren filtredir.

7- Band genişliği yalnızca filtrenin derecesi ile kontrol edilir.

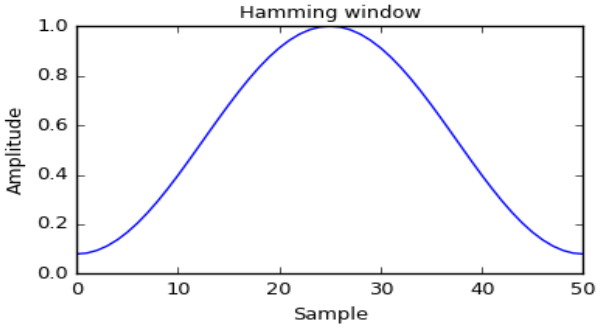
Çerçeveleme, konuşmanın durağan olmayan (istatistikleri zamanla değişen) yapısından dolayı ses sinyalini belirli çerçevelere (epoch) bölme ihtiyacı duyarız. Bu işlemde sinyalden elde ettiğimiz her bir çerçeveyi de alt pencerelere böleriz.

Böylelikle her bir pencere için durağana yakın istatislikler elde ederiz. Uygulamamızda 20 ms'lik (50 adet çerçeve = 1 saniye) çerçeve örnekleri alınmıştır. Şekil 2.3'de bu yapı görülmektedir.



Şekil 2.3 : Frame Blocking (Çerçeveleme).

Pencereleme, pencereleme aşamasında, alınan sinyal içerisindeki devamsız kısımların dikkate alınmaması ses tanıma için kritik bir eşiktir. Bu işlem pencereleme ile gerçekleşmektedir. Pencereleme, elde edilen ses sinyalinde spektral analiz yapabilmemizi sağlar. En yaygın kullanılan pencereleme metodu ise Hamming Penceresidir. Bunun sebebi ise, pencere yan loblarının sinyal örneklemesinin gürültü tabanının çok altında kalmasını gerektiren herhangi bir durum olmamasıdır. Bunu ise ideal dürtü tepkisinin düzgün bir şekilde kesilmesini ve daha iyi görünen bir frekans tepkisini sağlamak için giderek daha karmaşık kosinüs fonksiyonlarını kullanarak sağlamaktadır [13]. Şekil 2.4'de Hamming Penceresi gösterilmiştir.



Şekil 2.4 : Hamming Window [13].

Diğer pencereleme fonksiyonları ise sırasıyla Hanning, Blackman ve Diktörgen pencereleme yöntemleridir. Pencereleme tekniklerinin formüle dökülmüş hallleri aşağıda verildiği gibidir;

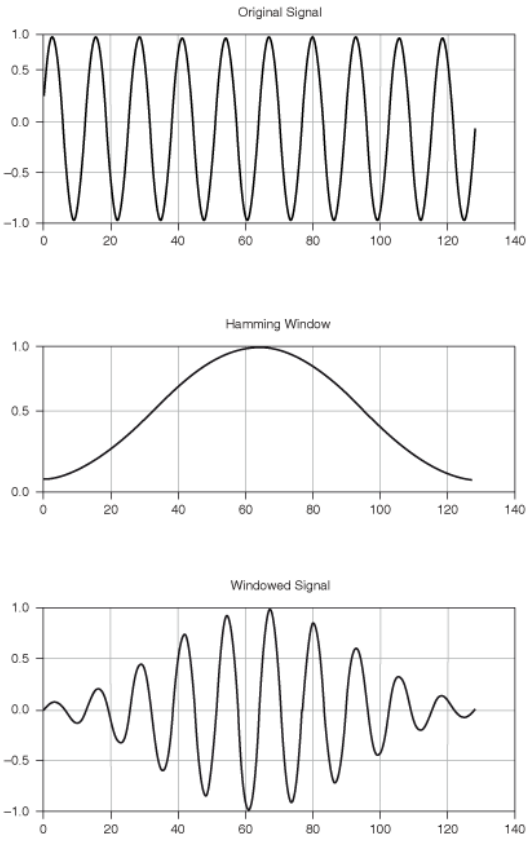
Hamming Pencereleme: $w(n) = [0.54 - 0.46*\cos(\frac{2\pi n}{N-1})], \quad 0 <= N - 1$ (2.2)

Hanning Pencereleme: $w(n) = [0.5(1 - \cos(\frac{2\pi n}{N-1}))], \quad 0 <= N - 1$ (2.3)

Blackman Pencereleme: $w(n) = [0.42 - 0.5\cos(\frac{2\pi n}{N-1}) + 0.08\cos(\frac{4\pi n}{N-1})], \quad 0 <= N - 1$ (2.4)

Dikdörtgen Pencereleme: $w(n) = 1, \quad 0 <= N - 1$ (2.5)

Şekil 2.5'de de görüleceği üzere Hamming penceresinin bir sinyale nasıl uygulanacağı görülmektedir.



Şekil 2.5: Pencereleme Metodu Uygulanmış Bir Sinyal.

Ayrık Zaman Fourier Dönüşümü (DFT):

$$x[n] \Leftrightarrow X(e^{-jw}) \text{ ve } X(e^{-jw}) = \sum_{-\infty}^{\infty} x[n]e^{-jwn} \quad (2.6)$$

Çerçevelere bölüp pencreleme uyguladığımız her bir parça için FFT dönüşümü sağlanır. Ana amacımız ise her çerçevenin güç spektrumunu hesaplayabilmektir.

Çünkü, gelen seslerin frekansına bağlı olarak farklı noktalardaki titreşimler insan vücudunda koklea (kulaktaki bir organ) tarafından motive edilir. Kokleadaki titreşen (küçük kılları sallayan) yere bağlı olarak, farklı sinirler, beyne belirli frekansların mevcut olduğunu bildirir. Çerçevede hangi frekansların mevcut olduğunu belirleyerek bizim için benzer bir iş gerçekleştirebilmek için güç spektrumunu doğru bir şekilde hesaplayabilmekten geçer. Her bir frame için güç spektrumu N= 512 olmak üzere aşağıdaki denklem ile hesaplanır;

$$P = \frac{FFT(xi)^2}{N} \quad (2.7)$$

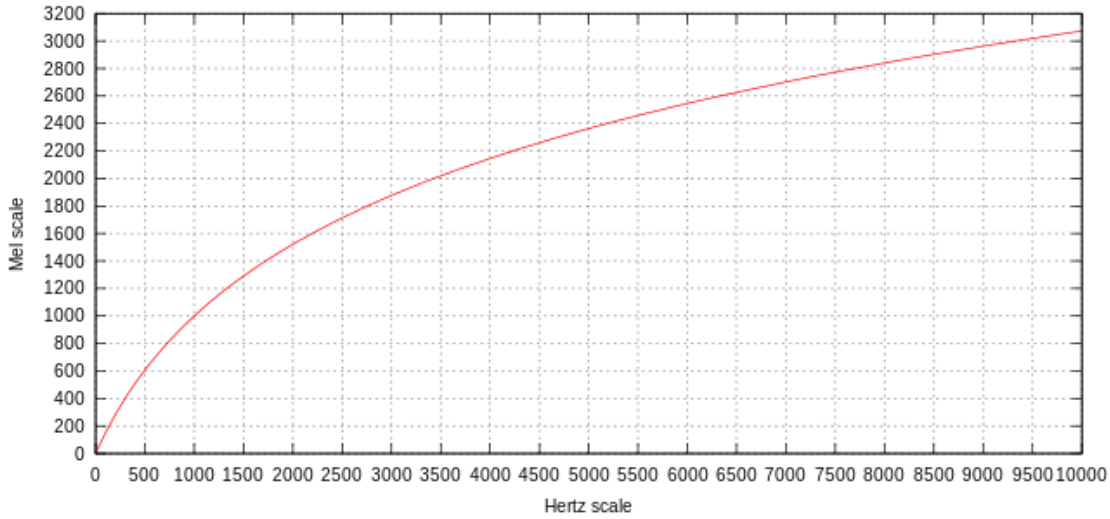
Fast Fourier Transform (FFT)**,** Ayrık Fourier Dönüşümü'nün (DFT) daha hızlı bir versiyonudur. FFT, DTF ile aynı amaç için bazı akıllı algoritmalar kullanır, ancak

işlemleri çok daha hızlı şekilde gerçekleştirir. Discrete Fourier Transform (DFT), frekans spektrum alanında oldukça önemli bir kavramdır. Çünkü zaman alanındaki ayrık bir sinyali frekans domenine çevirmek için bir araç olarak kullanılır. Bu dönüşüm olmadan bir sinyalin Fourier katsayılarını mikroişlemci veya DSP bazlı bir sistem vasıtasyla hesaplamak imkansı olacaktır. FFT'nin mantığı anlamak için aşağıda bazı denklemler verilmiştir; [13]

DFT ve FFT'yi basitçe karşılaştırmak gerekirse Ayrık Fourier Dönüşümü (DFT) olarak adlandırılan, zaman alanı sinyallerini frekans alanı bileşenlerine dönüştüren algoritmadır. DFT, adından da anlaşılacağı gibi, gerçekten ayrıktır; ayrık zamanlı etki alanı veri kümeleri ayrık frekans gösterimine dönüştürülür. Basit bir ifadeyle, zaman alanı gösterimi ile frekans alanı gösterimi arasında bir ilişki kurar. Hızlı Fourier Dönüşümü veya FFT, büyük dönüşümlerin hesaplama süresini ve karmaşıklığını azaltan bir hesaplama algoritmasıdır. Yani FFT sadece DFT'nin hızlı hesaplanması için kullanılan bir algoritmadır. Bir DFT zaman karmaşıklığında O(N^2) olarak gerçekleştirilebilirken, FFT zaman karmaşıklığını O($N\ logN$) sırasına göre azaltır.

Mel Filtre Uygulanması: Mel bir tonun algılanan frekansının ölçüsüdür. Bu ölçü tonun fiziksel frekansıyla lineer olarak örtüşmez. Çünkü insan duyma sistemi frekansları lineer olarak algılayamaz. Bundan dolayı mel skalası deneyi yapılmıştır. 1000 Hz'lik bir ton frekans olarak seçilmiş ve buna 1000 mels denilmiştir. Daha sonrasında ise algıladıkları frekans 2 katına çıkarılmıştır ve bu frekans 2000 mels olarak adlandırılmıştır. Devam eden süreçte bu deney frekasnının 10, 100, 0.5 vb katları için tekrarlanmıştır.

Bu sonuçlar sırasıyla 10000 mels, 100000 mels ve 500 mels olarak adlandırılmıştır. Bunun sonuunda ise gerçek fiziksel frekans (Hz) ile algılanan frekans (Mel) arasında bir eşleşme yapma imkânı doğmuştır. Bu eşleşmenin 1 Khz'in üstünde logaritmik altında ise lineer olduğu görülmüştür. [14]



Şekil 2.6 : Mel Skalası.

Formülasyon aşağıda verildiği gibidir:

$$\text{Fmel} = 2595 * log_{10}(1 + \frac{f_{hertz}}{700}) \quad (2.8)$$

Ya da

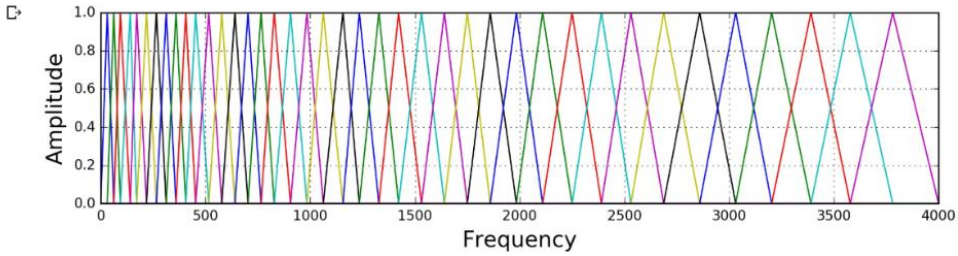
$$\text{Fmel} = 1127 * ln(1 + \frac{f_{hertz}}{700}) \quad (2.9)$$

olarak dönüşüm sağlanabilir.

Çoğu mel ölçekli formül 1000 Hz'de tam olarak 1000 mels verir. Kesme frekansı (örneğin 700 Hz, 1000 Hz veya 625 Hz), formülün normal formundaki tek serbest parametredir. Bazı erimeyen işitsel frekans ölçeği formülleri aynı formu kullanır, ancak çok daha düşük kopma frekansıyla 1000 Hz'de 1000 ile eşleme yapmak zorunda değildir; örneğin Glasberg & Moore'un (1990) ERB oranı ölçeği 228.8 Hz kırılma noktası, ve Greenwood'un koklear frekans yeri haritası (1990) 165.3 Hz kullanır. Mel skalası için diğer fonksiyonel formlar Umesh ve diğerleri tarafından araştırılmıştır; logaritmik bir bölgeye ve doğrusal bir bölgeye sahip geleneksel formüllerin, bu eğrilerden yaptıkları aşağıdaki ölçümler tablosuna dayanarak Stevens ve Volkmann'ın eğrilerinden ve diğer bazı formlardan gelen verilere uymadığına dikkat çekmektedir.

Spektral tahmini hala Otomatik Konuşma Tanıma (ASR) için gerekli olmayan birçok bilgi içermektedir. Özellikle koklea, birbirine yakın aralıklı iki frekans arasındaki farkı ayırt edemez. Bu etki, frekanslar arttıkça daha belirgin hale gelir. Bu nedenle, çeşitli frekans bölgelerinde ne kadar enerji bulunduğuna dair bir fikir edinmek için

periodogram kümelerini alıp toplarız. Bu, Mel filtre bankamız tarafından gerçekleştirilir: ilk filtre çok dardır ve 0 Hertz yakınında ne kadar enerji bulunduğunu gösterir. Frekanslar yükseldikçe, varyasyonlar konusunda daha az endişe duyduğumuz için filtrelerimiz genişler. Filtre bankası üçgen bant geçiren frekans karakteristiğine sahiptir ve bant genişliği sabit mel frekansı aralıklarıyla belirlenir. Filtre bankası 13 doğrusal aralıklı ve 27 logaritmik aralıklı filtreden oluşur. Şekil 2.7'de ve Şekil 2.8'de oluşturulan mel filtre bankası görseli ve band aralıkları görülmektedir.



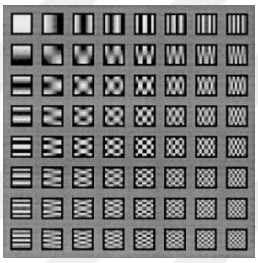
Şekil 2.7 : Colab Mel Filtre Bankası.

Index	Center Freq. (Hz)	BW (Hz)
1	100	100
2	200	100
3	300	100
4	400	100
5	500	100
6	600	100
7	700	100
8	800	100
9	900	100
10	1000	124
11	1149	160
12	1320	184
13	1516	211
14	1741	242
15	2000	278
16	2297	320
17	2639	367
18	3031	422
19	3482	484
20	4000	556
21	4595	639
22	5278	734
23	6063	843
24	6964	969

Şekil 2.8: Mel Skala Aralıkları.

Filtre bankası enerjilerini elde ettikten sonra, bunların logaritmasını alırız. Genellikle bir sesin algılanan ses seviyesini ikiye katlamak için 8 kat daha fazla enerji koymamız gerekir. Bu, sesin başlaması yüksekse, enerjideki büyük değişikliklerin o kadar farklı gelmeyebileceği anlamına gelir. Bu sıkıştırma işlemi özelliklerimizin insanların gerçekte duyduklarıyla daha yakından eşleşmesini sağlar. Logaritma, kanal normalizasyon tekniği olan cepstral ortalama çıkarmayı kullanmamızı sağlar. [15]

DCT Matris: Son adımda ise elde ettiğimiz logaritmik mel filtre bankası enerjilerinin DCT'sini hesaplamaktır. DCT (Discrete Cosine Transform) bir işaretin frekans bileşenlerine ayrılmasıdır. Bunun sebebi ise mel spektrumunun logaritması zaman bölgesine çevrilmesi olduğundan, frekans tanım bölgesine geçmek için DCT yönteminin seçilmesidir. Şekil 2.9'da DCT Matris uygulaması görülmektedir. Şekil 2.10'da Colab programi kapsamında elde edilen MFCC katsayılarını göstermektedir.



Şekil 2.9: DCT Matris Uygulaması [16].

$$c(n) = u_n * \sum_{k=0}^{K-1} logY_k * \cos(\frac{(2k+1)*pi*n}{2K}) \qquad (2.10)$$

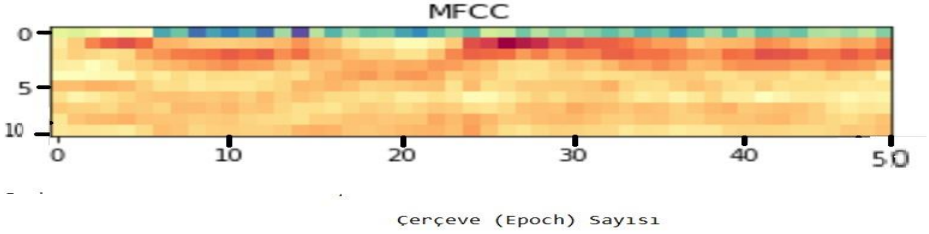
Burada,

n = mel frekans kepstral katsayısı indeksi

k = mel filtresi indeksi

K = toplam mel filtresi sayısı anlamına gelmektedir.

Y_K : Özgün ses sinyali

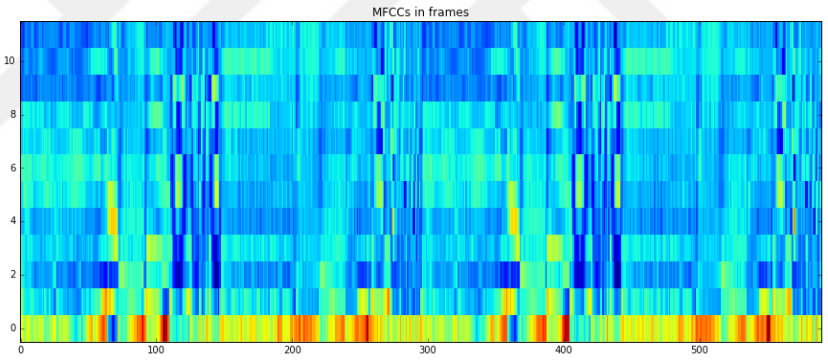


Şekil 2.10 : MFCC.

Yukarıdaki şekil 2.10'da yatay eksen, çerçeve (epoch) sayısını göstermektedir (20 ms'lik adımlarla, 1 saniyeye denk gelen 50 çerçeve bulunmaktadır). Dikey eksen ise MFCC katsayılarını belirtir. Renkler açığa gittikçe genlik düşer.

DCT matrisini elde ettikten sonra discrete bir konuşma sinyaline ait MFCC katsayılarını elde etmiş olduk. Fakat her MFCC katsayısının gücü aynı değildir, giderek azalmaktadır. Buyüzden MFCC katsayıları ile işlem yaparken ilk en güçlü 10 MFCC katsayı işleme alınır.

Spectogram Matrisin Oluşturulması: Spectogram matris bir sinyalin frekans spektrumunu görsel olarak elde ettiğimiz bir temsildir. DCT uygulaması da yapıldıktan sonra elde edilen MFCC katsayılarının en güçlü ilk 10 elemanını alındığını belirtmiştik. Buna ek olarak bu 10 elemanın her birisi 20 sn'lik pencereler ile 49 parçaya bölünmüş ve derin öğrenme aşamasında kullanacağımız modelin inputunu oluşturmuştur [17]. Şekil 2.11'de MFCC Spectogram matrisi katsayıları örnek görseli görülebilir.



Şekil 2.11: MFCC Spectogram Matrisi Katsayıları.

2.1.2 Kullanılan Programlar

Bu başlık altında sinyal işleme aşamasında hangi programların kullanıldığını özetlenmiştir. Sinyal işleme sürecinde PC ortamında Colab, MATLAB ve Praat programları; gömülü programlama tarafında ise NXP firması tarafından geliştirilen MCUXpresso platformunda C ve C++ dilleri cross compile olarak kullanılmıştır.

2.1.2.1 PC'de Sinyal İşleme Ortamları

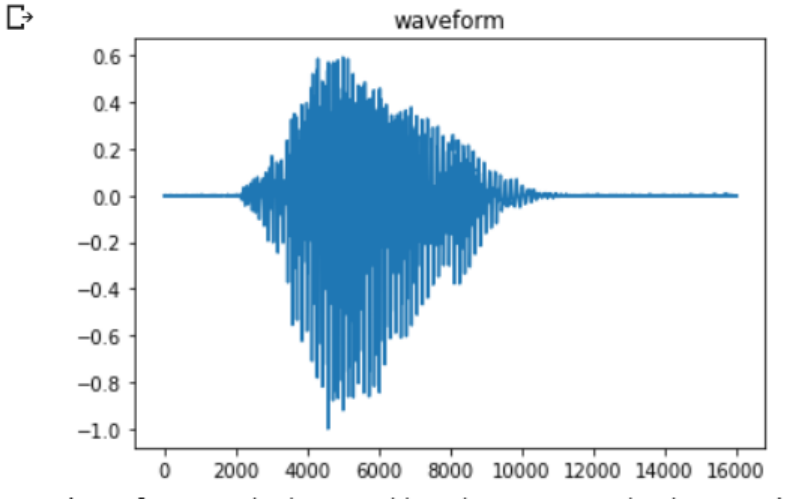
Bu başlık altında Colab, Matlab ve Pratt programları kapsamında yapılan çalışmalara yer verilmiştir.

Colab: Bu program Google'ın geliştiricilere sunduğu ücretsiz bulut servisidir. Pyton programlama dilinde uygulama geliştirmemize olanak sağlar.

İleri ki bölümlerde bahsedeceğimiz Keras, Tensorflow gibi kütüphanleride kullanılarak derin öğrenme uygulamaları geliştirebiliriz. Colab'ın yararlandığımız en büyük özelliği ise kullanıcıya ücretsiz GPU sağlamasıdır. Şekil 2.12 ve Şekil 2.13'de ise çalışma ortamı olan Colab ve elde edilen bir ses analiz görseli görülmektedir.

Şekil 2.12 : Google Colab.

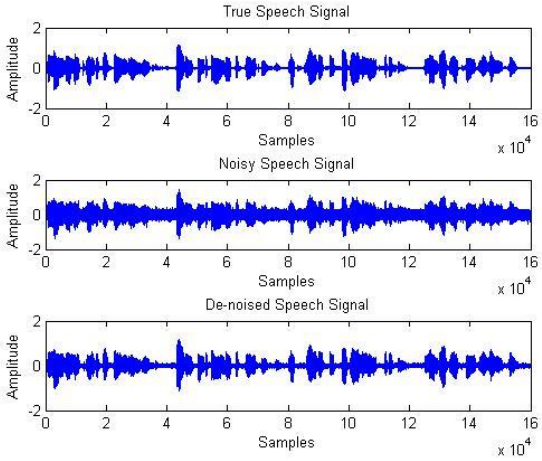
```
[ ]  plt.figure()
     plt.plot(X_train_waveform[0])
     plt.title("waveform")
     plt.show()
     X_train[0]
```



Şekil 2.13: Google Colab ses sinyali grafiği.

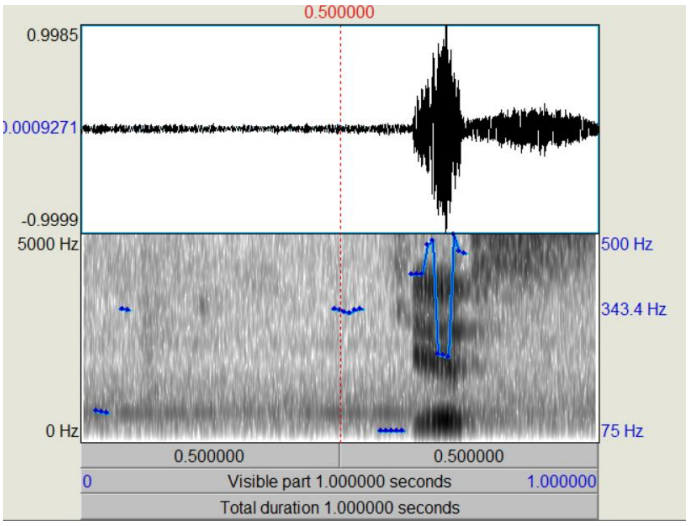
MATLAB: Sinyal işleme uygulamalarında sıklıkla kullanılan ve Simulink ortamında DSP işlemlerin simülasyon kolaylığı sağladığı için tezde kullanılmıştır.

Ayrıca kullanıcıya sağlamış olduğu hazır fonkisyon olanklarıda sinyal işleme projelerinde oldukça kullanışlıdır. Şekil 2.14'de ise FIR filtre uygulanmış bir sinyal örneği verilmiştir.



Şekil 2.14 : FIR filtre ile gürültü optimizasyonu.

Praat: Praat, Amsterdam Üniversitesi Fonetik Bilimler Bölümünden Paul Boersma ve David Weenink tarafından geliştirilen ücretsiz bir yazılımdır. Praat mono ve stereo sesler kaydetme ve yoğunluk, ses yüksekliği, süre veya formantlarla ilgili sesleri düzenleme ve analiz etme imkânı sunar. Ayrıca, ses nesnelerini yazıya dökmek ve açıklama eklemek için TextGrid-Objects olarak adlandırılabilir. Yazılım çalışmaları öncesi için bir ses formatındaki sinyalleri iyi anlayabilmek için kullanılabilecek güzel bir uygulamadır. Tez kapsamında konuşma sinyallerinin analizi ve anlaşılması amacıyla kullanılmıştır. Şekil 2.15 ve Şekil 2.16'da ise Pratt programı çalışma ortamı gösterilmiştir



Şekil 2.15 : PRAAT ile ses analizi.



Şekil 2.16: Praat giriş menüsü.

2.1.2.2 Gömülü Sistemlerde Ses işleme Ortamı

PC ortamında ilgili programlar ile analizler tamamlandıktan sonra ses sinyalinin işlenmesi aşaması gömülü sistemlerde gerçekleştirilmesi gerekmektedir. Bu konuya ileriki bölümlerde daha detaylı şekilde bakılacaktır. Bu başlık altında ise ön bilgi olması bakımından hangi program ve ide'nin kullanlıdığı bilgisi verilecektir. Öncelikle gömülü sistem olarak tercih ettiğimiz geliştirme kartı NXP ailesine ait Cortex M7 işlemci mimarisine ve 600 MHZ saat frekansına sahip bir işlemcidir. Sinyal işleme yazılım C / C++ dilinde cross compile olarak ve NXP MCUXpresso ide'si kullanılarak yazılmıştır. Şekil 2.17'de ise MCUXpresso ide gömülü sistem ortam derleyicisi görseli verilmiştir.



Şekil 2.17 : MCU Xpresso IDE.

2.2 Ses işleme çıktıları

Bölüm 4 "Ses İşleme" başlığı altında bir analog sinyalin bileşenlerinin neler olduğu, dijital sinyale dönüştürülme adımları, bu adımlarda kullanılan matematiksel fonksiyonları, elde edilen dijital işaret üzerinden derin öğrenme metodunda giriş olarak verdiğimiz spektogram matrisinin elde edilişini ve özellik verilerini çıkarma yönteminin nasıl uygulandığı adım adım gösterilmiş oldu. Bu alt başlıklardan sonra ise bütün bu işlemlerin hangi programlar vasıtasıyla yapıldığı verilmeye çalışıldı. 5. Bölüm olan "Derin Öğrenme" başlığı altında ise elde edilen bu dataların nasıl anlamlandırıldığını anlatılmaya çalışılacaktır.

3 DERİN ÖĞRENME

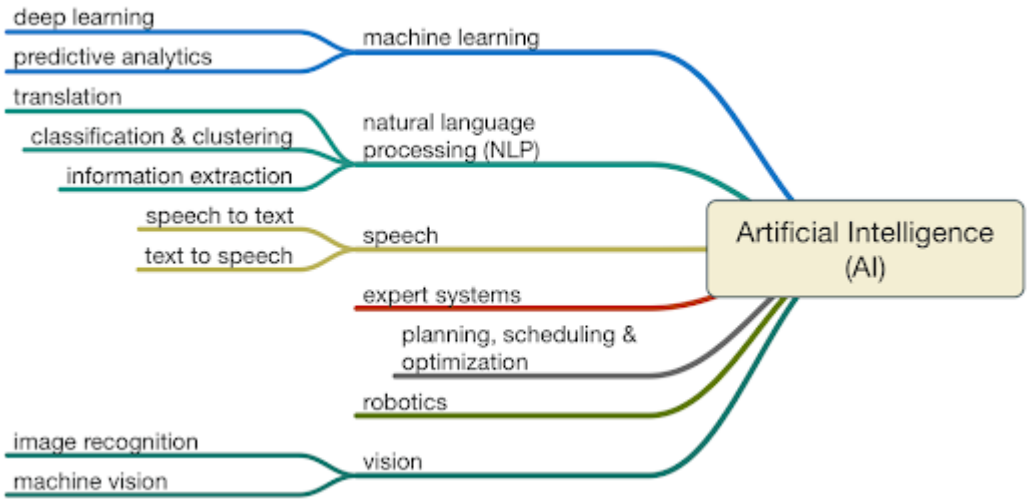
Derin öğrenme bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Derin öğrenmeyi ve kullanım alanlarını iyi anlayabilmek için bazı terimlere hâkim olunması gerekmektedir. “Derin Öğrenmeye Giriş” alt başlığında genel kavramlar ve tanımlar üzerinde durulacaktı.

3.1 Derin Öğrenmeye Giriş

Yapay zekâ (AI), insanlar gibi düşünmek ve davranışlarını taklit etmek için programlanmış bir makinede insan zekasının simülasyonunu ifade eder. En önemli özelliklerinden biri, belirli bir hedefe ulaşmak için en iyi fırsat olan rasyonalizasyon ve harekete geçme yeteneğidir. Yapay zekâ araştırması, bilgisayar bilimi, psikoloji, felsefe, sinirbilim, bilişsel bilim, dilbilim, yöneylem araştırması, ekonomi, kontrol teorisi, olasılık, optimizasyon ve mantık gibi birçok alandan araçlar ve tahminler kullanır. Yapay zekâ araştırmasının nihai hedefi, öğrenebilen, sorunları çözebilen ve mantıklı düşünebilen bilgisayar programları oluşturmaktır. [18]

Makine öğrenmesi, data set biriktirerek, bu biriken data seti temel alan istatistik kullanan bir modeli algoritma ile eğiten, hızlı bir sorun çözme aşaması olarak tanımlamayabiliriz. En genel anlamda, makine öğrenimi, makinelere kendi başlarına nasıl öğreneceklerini öğretme bilimidir. Bize sağladığı en büyük avantajlardan biri, makinenin sıkışmadan yüksek frekanslı tekrarlayan işleri yüksek hassasiyetle tamamlayabilmesidir. Makine öğrenimi algoritmaları genellikle denetimli veya denetimsiz olarak sınıflandırılır. Denetimli algoritmalar, girdi ve gerekli çıktıları sağlamak ve algoritma eğitimi sırasında tahmin doğruluğu hakkında geri bildirim sağlamak için veri bilimcilerinin veya makine öğrenme becerisine sahip veri analistlerinin kullanılmasını gerektirir [18]. Derin öğrenme, insan beyninin çalışmasını taklit eden yapay bir zekâ işlevidir. Veri işleme ve karar verme için kullanılabilecek modeller oluşturmak için tasarlanmıştır.

Çok büyük bir sinir ağı ve erişilebilen büyük miktarda veri gerektirir. Makine öğrenimi daha basit kavramlar kullanırken derin öğrenme, insanların nasıl düşündüklerini ve öğrendiklerini taklit etmek için tasarlanmış yapay sinir ağları ile çalışır. Sinir ağları, tıpkı insan beyninin nöronlardan oluştuğu gibi katmanlardan oluşur. Tek tek katmanlardaki düğümler bitişik katmanlara bağlanır. Ağın sahip olduğu katman sayısına göre daha derin olduğu söylenir. İnsan beynindeki tek bir nöron, diğer nöronlardan binlerce sinyal alır. Yapay bir sinir ağında, sinyaller düğümler arasında seyahat eder ve ilgili ağırlıkları atar. Daha ağır ağırlıklı bir düğüm, bir sonraki düğüm katmanı üzerinde daha fazla etki gösterecektir. Son katman, bir çıktı üretmek için ağırlıklı girdileri derler. Derin öğrenme sistemleri güçlü bir donanım gerektirir, çünkü büyük miktarda veri işlenir ve birkaç karmaşık matematiksel hesaplama içerir. Bununla birlikte, bu tür gelişmiş donanımlarla bile, derin öğrenme eğitimi hesaplamaları haftalar sürebilir. Derin öğrenme sistemleri, doğru sonuçlar elde etmek için büyük miktarda veri gerektirir; buna göre, bilgi büyük veri setleri olarak beslenir. Verileri işlerken yapay sinir ağları, verileri oldukça karmaşık matematiksel hesaplamaları içeren bir dizi ikili doğru veya yanlış sorudan alınan cevaplarla sınıflandırabilir. Örneğin, bir yüz tanıma programı, yüzlerin kenarlarını ve çizgilerini, daha sonra yüzlerin daha önemli kısımlarını ve son olarak yüzlerin genel temsillerini tespit etmeyi ve tanımayı öğrenerek çalışır. Zamanla, program kendini eğitir ve doğru cevap olasılığı artar. Bu durumda, yüz tanıma programı yüzleri zamanla doğru bir şekilde tanımlayacaktır. [18] Şekil 3.1'de yapay zeka ağaç şematiği yer almaktadır.



Şekil 3.1 : Yapay Zekâ Ağaç Şematiği [19].

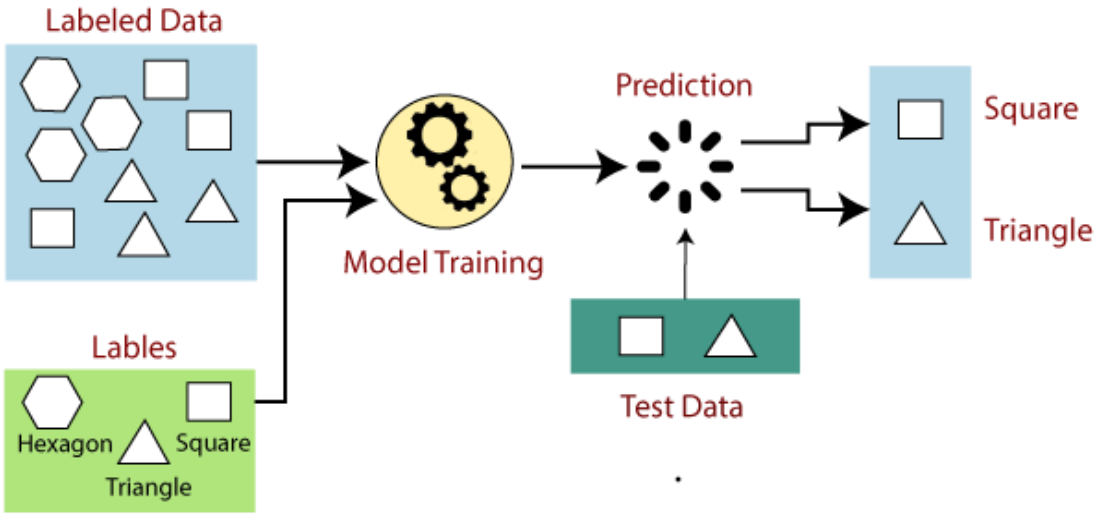
Yapay zekâ (AI), makine öğrenmesi, derin öğrenme arasındaki farkları ve ilişki ağının elde edilmesinin ardından “Derin Öğrenme” ile ilgili kavramlar üzerinde durulmaya çalışılmıştır.

Genel olarak literatür içerisnde derin öğrenme başlığı altında 7 mimari üzerinde durulur. Bu 7 mimari ise 3 ana başlık içerisinde ayrılırlar. Bu dağılımı çizelge 3.1’de görebiliriz.

Çizelge 3.1 : Derin Öğrenme Mimarileri.

Supervised Learning	**Unsupervised Learning**	**Reinforcement Learning**
Feed Forward Neural Networks	Autoencoder	Network for action, values and modes
Convolutional Neural Networks	Generative Adversarial Network	
Reccurent Neural Networks		
Encoder – Decoder Architecture		

Konuya daha detaylı bir giriş için Supervised Learning ve Unsupervised Learning kavramları tanımlanması gerekmektedir. Şekil 3.2’de örnek görsel görülmektedir.



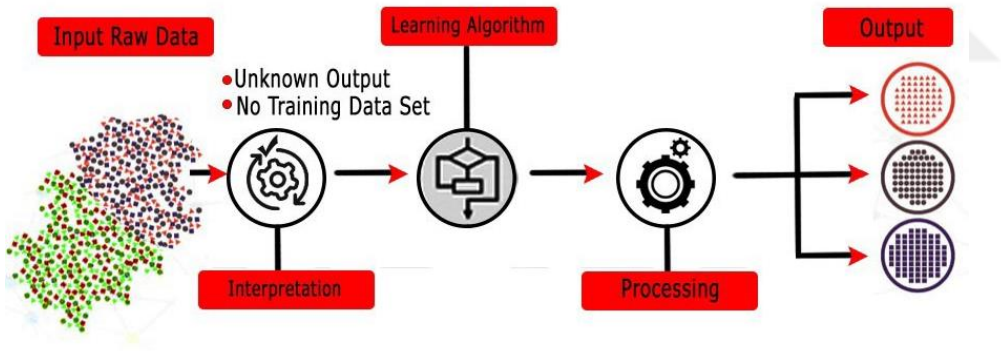
Şekil 3.2 : Supervised Learning Örnek Görsel.

Supervised Learning (SL): Eğitim dataları üzerinden bir fonksiyon üreten makine öğrenme tekniğine verilen isimdir.

Yani (işaretlenmiş data -labelled data) ile istenen arasında eşleşme yapan bir fonkiyon üretir. Eğitim verisi ise hem girdilerden hem de çıktılardan oluşur. Eğitim verisi hem girdilerden hem de çıktılardan oluşmaktadır.

Supervised Learning (SL): Eğitim dataları üzerinden bir fonksiyon üreten makine öğrenme tekniğine verilen isimdir. Yani (işaretlenmiş data -labelled data) ile istenen arasında eşleşme yapan bir fonkiyon üretir. Eğitim verisi ise hem girdilerden hem de çıktılardan oluşur. Eğitim verisi hem girdilerden hem de çıktılardan oluşmaktadır.

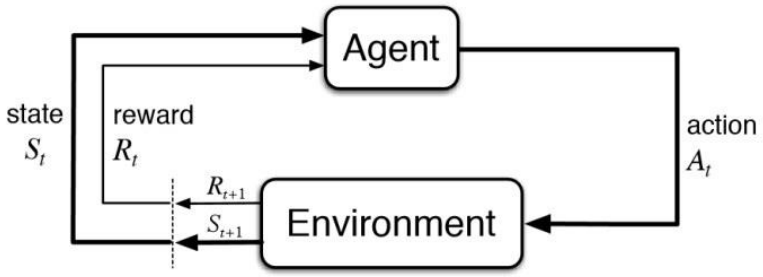
Unsupervised Learning (UL): Bu yöntemde ise işaretlenmemiş veri üzerinden bilinmeyen bir yapıyı tahmin etmek için fonkiyon kullanan makine öğrenmesi tekniğidir. Burada girdi verisinin hangi sınıfa ait olduğu belirsizdir. Bu bölümdeki algortimalar verileri gruplandırarak yeni veriyi en uygun gruba yerleştirmeye çalışır. Eğitim verisi olmadığı için uygulaması kolaydır. Şekil 3.3'de ilglili örnek gösterilmiştir.



Şekil 3.3 : Unsupervised Örnek Görsel.

Reinforcement Learning (RL): Bu, bir nesnenin ortamdaki en yüksek ödül miktarına ulaşmak için gerçekleştirmesi gereken eylemleri ele alan, davranışçılıktan esinlenen bir makine öğrenme yöntemidir. Bu yöntem insanların en basit şekilde öğrenme yöntemlerini simüle eder. İnsanlar edindikleri bilgileri kullanabildiklerinde veya daha yaygın süreçlerle öğrenebildiklerinde, gerçek hayatta doğduğumuz andan (ister kendimiz ister çevremizdekiler olsun) öğrenmeye başlarız, çevre ile etkileşime girer ve bu etkileşimlerin sonuçlarını gözlemlerizAynı yöntem makine öğrenmesi için de geçerlidir.

Özetle, sistem oturum açma durumunu izler. Duruma göre işlemler gerçekleştirir, alacağı ödülleri veya cezaları kontrol eder ve alınan ödüller veya para cezalarına göre öğrenme katsayısını günceller. Şekil 3.4'de örnek görsel yer almaktadır.



Şekil 3.4 : Reinforcement Learning Örnek Görsel.

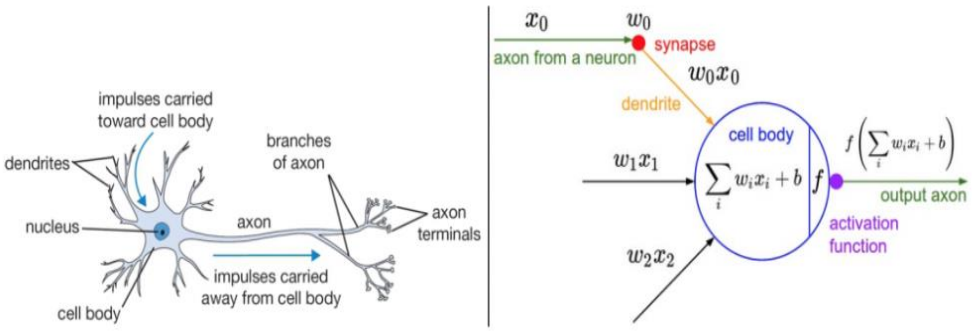
SL, UL ve RL yöntemlerinin özelliklerinin ve farklarının bilinmesi; projede ki amaç olan sesli komut tanıma uygulaması için doğru yöntemin seçilmesi büyük önem taşımaktadır. İlk başta sesli komut tanımada etiketlenmiş kelimeler mevcuttur. Yani sistem belirli kelimelere tepki vermeli ve diğer kelimeler için ise uykudan uyanmamalıdır. Dolayısıyla sisteme bir input vasıtasıyla tanıyacağı kelimeleri vermeli, bir dataset yardımıyla eğitmeli ve test datasıyla öğrenme katsayısını arttırmalıyız. Bu algortima ise bizi "Supervised Learning" metodu altında araştırma yapmaya yöneltmiştir. "Supervised Learning'e Giriş" başlığı altında SL metodlarını inceleyecek ve amaçladığımız sistem için en doğru modeli geliştireceğiz.

3.1.1 Supervised learning'e giriş

Bu sistemler, bir giriş verisinin etiketler vasıtası ile eğitildiği ve test dataların kullanılarak input'un doğru etiket ile etiketlenme ihtimalinin arttırıldığı sistemlerdir. "Supervised Learning'e Giriş" başlığı altında bu 4 yöntem incelenecek ve sesli komut tanıma projesi seçilen yöntem ve yapılan çalışmalar anlatılmaya çalışılacaktır. Fakat bu 4 alt başlık altında çalışma yapabilmemiz Artificial Neural Networks yani Yapay Sinir Ağları kavramlarına hâkim olmaktan geçecektir.

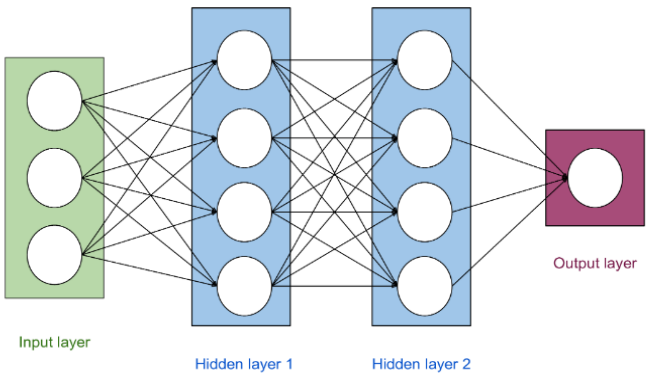
3.1.1.1 Yapay sinir ağları (artificial neural networks (ANN))

Yapay sinir ağının genel bir tanımı yapılması gerekirse; yapay sinir ağı, insan beyninin çalışma ve düşünebilme yeteneğinden yola çıkılarak oluşturulmuş bir bilgi işlem teknolojisidir. Yapay Sinir Ağları, en basit tanım ile, biyolojik sinir ağlarını taklit eden bilgisayar programları olarak adlandırılabilir. [19] Şekil 3.5'de genel bakoş şematiği görülmektedir.



Şekil 3.5 : Sinir Ağlarına Genel Bakış [20].

Bir yapay sinir hücresi 3 ana bölümden oluşur. Bunlar giriş katmanı (input layer), gizli katmanlar (hidden layers) ve çıkış katmanı (output layer) olarak sıralanabilir. Şekil 3.6'da katman yapısı görsel olarak yer almaktadır.

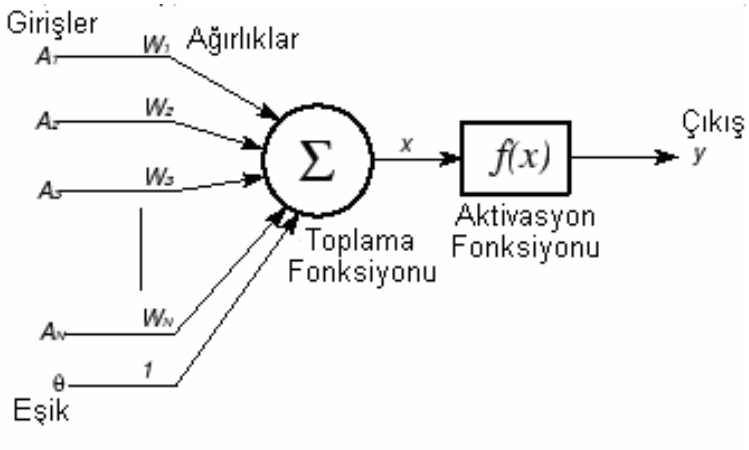


Şekil 3.6 : ANN Katman Yapısı [20].

3.1.1.2 Yapay sinir ağlarının eğitilmesi ve sınıflandırma

Yapay sinir ağlarının eğitilmesinde amaç hesaplanan değer ile hedef değer arasındaki hatayı minimuma indirmek ve girdiyi doğru etiket ile etiketleyebilmektir. Hatanın minimize edilmesi için de ağırlık değerlerinin hataya göre update edilmesi lazımdır. Eğitim aşamasında her adımda hata hesaplanırken ağırlar güncellenmektedir.

Böylece en doğru sonuca ulaşılmaya çalışılmaktadır. Eğitim adımları ve kullanılan fonkisyonlar aşğıda adım adım verilmektedir. [21]



Şekil 3.7 : Yapay Sinir Ağları Detaylı Mimarisi.

Model eğitimi aşağıdaki gibi bölerek inceleyeceğiz:

1) Başlangıç, bütün ağırlıkların [-1 ile +1] arasında rastgele değerlerle ilk değer verilmesi

2) Eğitim döngüsü: Tüm veri kümesini güncellleme adımı

* Feed – Forward (İleri beslemeli): güncelleme

* Calculate Error (Hata hesaplama)

* Back propagation (Geri yayılım mekanizması) : eğitimi

* Ağırlık ve bias değerleninin güncellenmesi

Feed – Forward (İleri beslemeli) : güncelleme

$$O_m^{l+1} = \Omega\left(\sum w_{m,n}^{l} * O_n^{l} + b^{l}\right) \quad (3.1)$$

İlk katmandan başlayarak sinir ağında ilerleyeceğiz. Fonkisyonda m, değerini hesaplamak istedğimiz nöronun pozisyonu belirtir. O bir nöron ve w ise m ve n nörolarını bağlayan ağırlık katsayısını ifade eder. b bias l katmanındaki bias değerini temsil eder. Toplama fonksiyonunun verdiği değeri işleyerek ağın çıktısını üreten fonksiyon olan Ω (net fonksiyonu ile) çarpılır. Bu fonksiyonlar genellikle yüksek değer üretmezler.

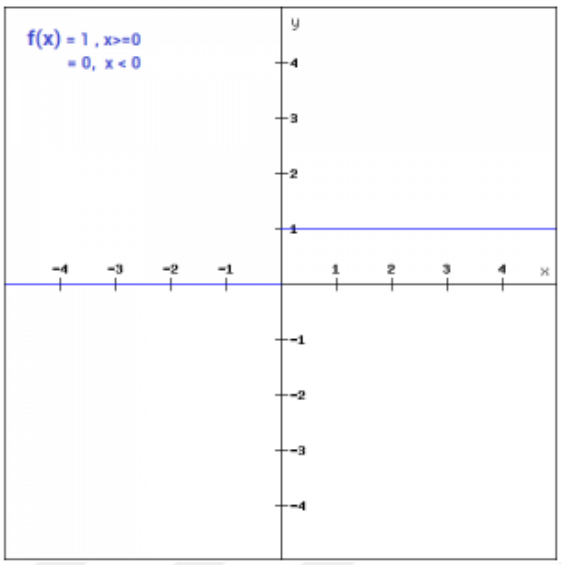
Aktivasyon fonksiyonlar aşağıdaki gibidir:

Birim Basamak Fonksiyonu (Binary Step Function) :

$$f(x) = 1, x >= 0$$

$$f(x) = 0, x < 0 \quad (3.2)$$

Türevi alınmasının bir şey ifade etmemesinden dolayı ara katmanlarda kullanılamaz, sadece çıkış katmanında kullanıma uygundur. Şekil 3.8'de yer alan birim bamak fonksitonunu ele alalım.

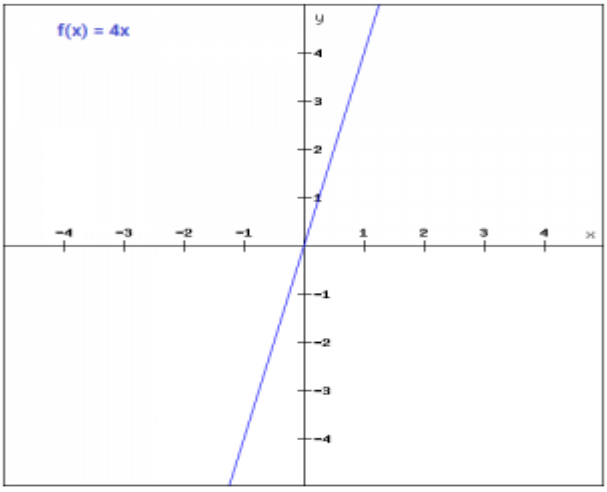


Şekil 3.8 : Birim Basamak Fonksiyonu [22].

Lineer Fonksiyon (Linear Function) :

$$f(x) = ax, a \in Z \quad (3.3)$$

Türevi alınabilir fakat türevi sabit bir değer üretir. Dolayısıyla geriye yayılım algoritmasında giriş değerinin hiçbir etkisi kalmaz. Bu fonksiyon da basamak fonksiyonu gibi sadece çıkışta kullanılabiliriz.



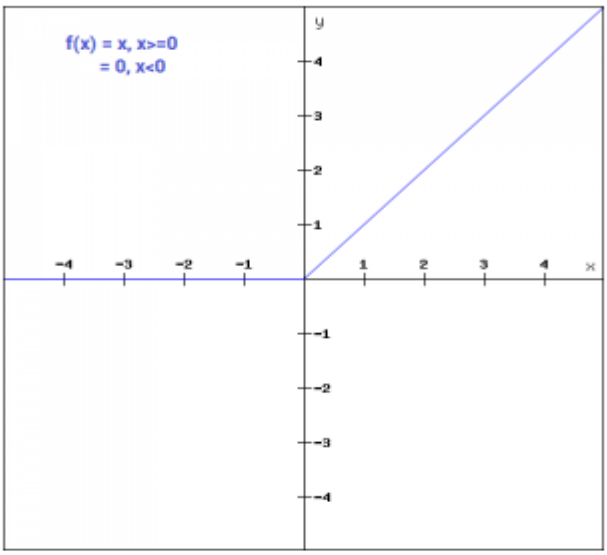
Şekil 3.9 : Lineer Foksiyon [22].

ReLU fonksiyonu:

$$f(x) = x,\ x > 0$$

$$f(x) = 0,\ x <= 0 \qquad (3.4)$$

Açılımı Rectified Linear Unit yani Doğrultulmuş Doğrusal Birim olarak geçmektedir. ReLU işlevinin kendisi ve türetilmiş işlevleri monotonik işlevlerdir. ReLU, herhangi bir negatif değeri sıfıra döndürür. Tüm negatif değerleri 0'a dönüştürmek, modelin doğru şekilde çalışmasını önleyecektir. Bu nedenle, model negatif değerler kullanılarak eğitilemez.

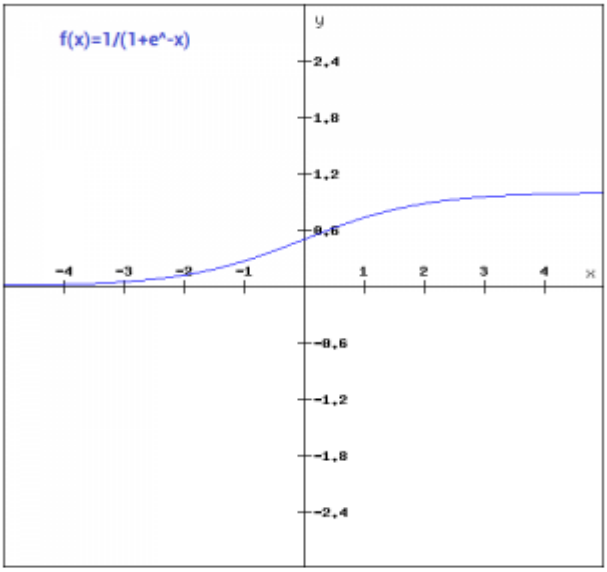


Şekil 3.10 : ReLu Fonksiyonu [22].

Sigmoid Function:

$$f(x) = \frac{1}{(1+e^{-x})} \tag{3.5}$$

En yaygın olarak kullanılan doğrusal olmayan aktivasyon işlevlerinden biridir. Sigmoid, değerleri 0 ile 1 arasında dönüştürür. Burada dikkat çekici olan nokta, ikili adım ve doğrusal fonksiyonların aksine, sigmoid'in doğrusal olmayan bir fonksiyon olmasıdır. Bu aslında-aktivasyon fonksiyonu olarak sigmoid fonksiyona sahip çoklu nöronlara sahip olduğumda, çıkışın da doğrusal olmadığı anlamına gelir.

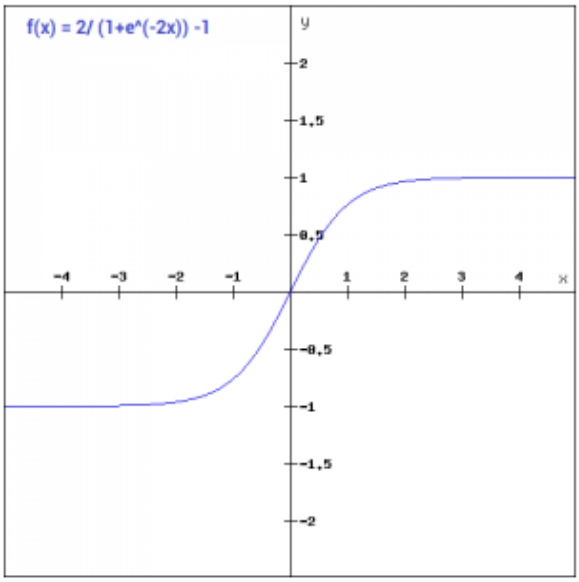


Şekil 3.11 : Sigmoid Fonksiyonu [22].

Tanh Fonksiyonu:

$$\tanh(x) = 2*\text{sigmoid}(2x) - 1 \tag{3.6}$$

Aslında Tanh fonksiyonu sigmoid işlevine çok benzer. Tek fark, orijin etrafında simetrik olmasıdır. Bu durumda değer aralığı -1'den 1'e kadardır. Dolayısıyla, sonraki katmanlara girişler her zaman aynı işaretten olmayacaktır.



Şekil 3.12 : Tanh Fonksiyonu [22].

Calculate Error (Hata hesaplama):

Hata, eğitim döngüsünün her yinelemesinde hesaplanmak zorundadır.

$$E = \frac{1}{2} \sum (target - output)^2 \qquad (3.7)$$

Yapay sinir ağındaki çıktı ile gerçek etiket veya hedef değeri arasındaki karesel farkların toplamı bize hata veya maliyet fonksiyonunu verir.

Back – Propagation (Geri Yayılım Mekanizması) :

Konu başında yapay sinir ağlarını tasarlarken, başlangıç değeri olarak ağırlıkların [-1 ile +1] değerleri arasında başlangıç değerleri veridliğini belirtmiştik. Ancak, başlangıçta bazı ağırlık değerleri seçtik, ancak model çıktımız gerçek çıktımızdan çok farklı, yani hata değeri çok büyük, işte bu sorunu çözmek için ağdaki ağırlıkların her birini güncellemek ve bu sayede gerçek çıktının hedef çıktıya daha yakın olmasına neden olmasını sağlamaktır. [23]

Bunu sağlamak içinse her bir ağırlığa göre hatanın kısmi türevini hesaplayacağız. Burada ise zincir kuralı devreye girer:

$$\frac{\partial E_{total}}{\partial w_{m,n}} = \frac{\partial E_{total}}{\partial out_{a_n^l}} * \frac{\partial out}{\partial net_{a_n^l}} * \frac{\partial net_{a_n^l}}{\partial w_{m,n}} \qquad (3.8)$$

Hatanın türevi:

$$\frac{\partial E_{total}}{\partial out_{a_n^l}} = -(x_n^l - a_n^l) \quad (3.9)$$

Elimizdeki bütün türevi alıyoruz:

$$\frac{\partial E_{total}}{\partial out_{a_n^l}} = -(x_n^l - a_n^l) * \Omega * (a_n^l) * a_n^{l-1} * a_m^{l-1} = \delta_n^l * a_m^{l-1} \quad (3.10)$$

Kullanılan aktivasyon fonksiyonu sigmoid fonksiyonuysa çıkacak olan türev:

$$\Omega'(x) = \Omega(x)(1 - \Omega(x)) \quad (3.11)$$

Daha sonra, sinir ağından ileri beslemenin (çıkışlardan girişlere) ters yönünde ilerleyeceğimiz h hata sinyali adı verilen yeni bir önemli terim oluşturmalıyız. Hatayı hesaplamak istediğimiz ağırlığın önceki nöronuna bağlı olmayacaktır.

$$\delta_n^l = -(x_n^l - a_n^l) * \Omega'(a_n^l) \quad (3.12)$$

Her yeni türev şimdi önceki nöronlara bağlı olacaktır, böylece hesaplamalarımız değişecektir.

$$\delta_n^{l+1} = \sum_{m=0} \delta_n^l * w_{m,n}^{l+1} \quad (3.13)$$

En son elde ettiğimiz denklem:

$$\frac{\partial E_{total}}{\partial w_{m,n}} = (\sum_{m=0} \delta_n^l * w_{m,n}^{l+1}) * \Omega' * a_n^{l-1}) \quad (3.14)$$

Her bir bias değerine göre hatayı hesaplamak istiyorsak, önceki nöron değerine bağlı olmayacaktır çünkü bir olacaktır, bundan dolayı aşağıdaki formülü elde edceğiz:

$$\frac{\partial E_{total}}{\partial b_m^l} = \delta_n^l \quad (3.15)$$

Genel Formülün Elde Edilişi:

Tüm bu hesaplamalarla, hatayı aşağıdaki formülle azaltmak için her bir ağırlığın varyasyonunu elde edebileceğimiz sonucuna varabiliriz:

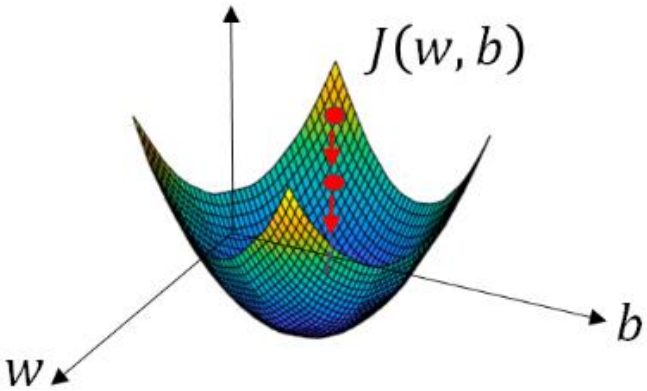
$$\frac{\partial E_{total}}{\partial w_{m,n}} = \delta_n^l * a_m^{l-1} \quad (3.16)$$

Bias için elde ettiğimiz çıkarım:

$$\frac{\partial E_{total}}{\partial b_m^l} = \delta_n^l \quad (3.17)$$

Ağırlıkların Güncellenme Algoritması :

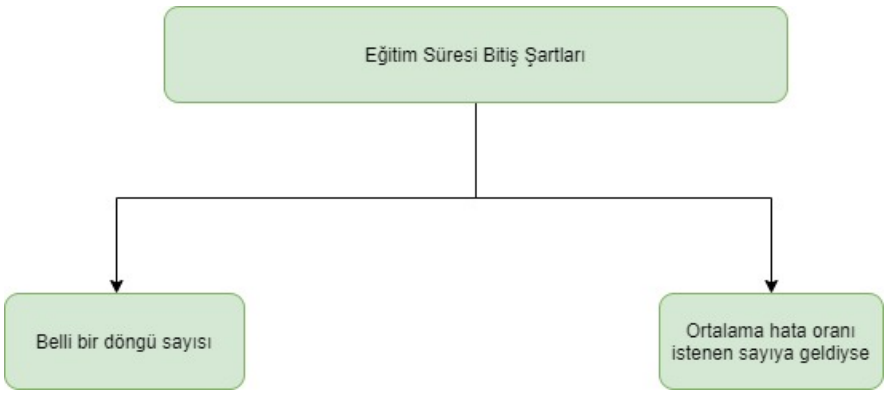
Burada maliyeti minimize etmek için en uygun ‘w’ ve ‘b’ değerlerini bulmaya çalışacağız. Bunu da gradient descent (gradient düşüşü) denilen bir yöntem ile gerçekleştireceğiz. Gradyan inişi, makine öğrenmesi alanında popüler bir yöntemdir, çünkü makine öğrenmesinin amaçlarından biri, eğitim verisi göz önüne alındığında, en yüksek doğruluğu bulmak veya hata oranını en aza indirmektir. Gradyan inişi “maliyet fonksiyonunu“en aza indirgeyerek asgari hatayı bulmak için kullanılır. [24]



Şekil 3.13 : Graident Descent 3 boyutlu görsel.

Gradient descent algoritması denklemin local minumuma yakınsaması için birinci türevi kullanır. Hata fonksiyonunun lokal minimumu bu algortima ile hesaplanır. Bu algortimada Learning rate (Lr) yani öğrenme katsayısı çok kritik bir rol oynamaktadır.

Eğer bu katsayı optimum olarak seçilmez ise yani olmasından daha küçük seçilirse epoch yani tekrar sayısı artar bu da maaliyeti arttırmaktadır. Büyük katsayılar kullanılırsa ise az epoch’da ulaşabilir fakat bu da yanlış noktaya yakınsamaya sebep olur. Yeni ağırlık değerinin hesaplanmasında öğrenme katsayısına ek olarak bir önceki ağırlık değeri ve momentum değeri de işleme katılmaktadır. Bu sayede ağırlıklar istenen optimum değere ulaşır. Şekil 3.14’de eğitim bitiş şartları yer almaktadir.



Şekil 3.14 : Yapay Sinir Ağları Kullanılan Bir Sistemin Eğitiminin Sonlanması.

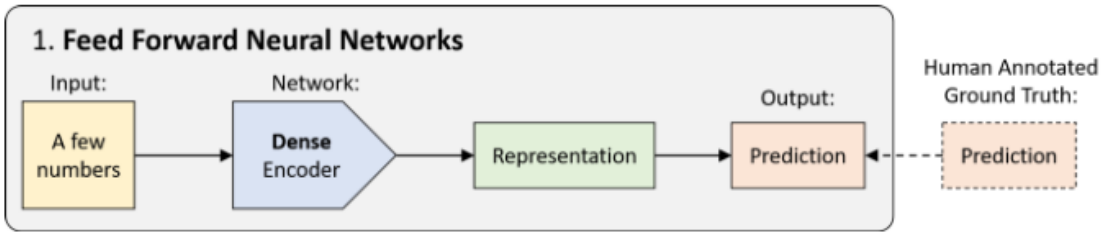
Her eğitimdeki tüm ağ ağırlıklarını ve bias fonksiyonlarını güncellemek için kullanılan fonksiyon aşağıdaki gibidir:

$$w_{m,n}^{l} = w_{m,n}^{l} - l_r * \frac{\partial E_{total}}{\partial w_{m,n}} \quad (3.18)$$

Yapay sinir ağlarının eğitim aşamasını bitirmiş olup bundan sonraki başlık altında yapay sinir ağı modelleri, bu modellerin birbirinden farkları ve çalışmamız için seçilecek olan modele karar verme aşamasına detaylıca yer verilmeye çalışılacaktır.

3.1.1.2.1 Feed forward neural networks (FFNNs)

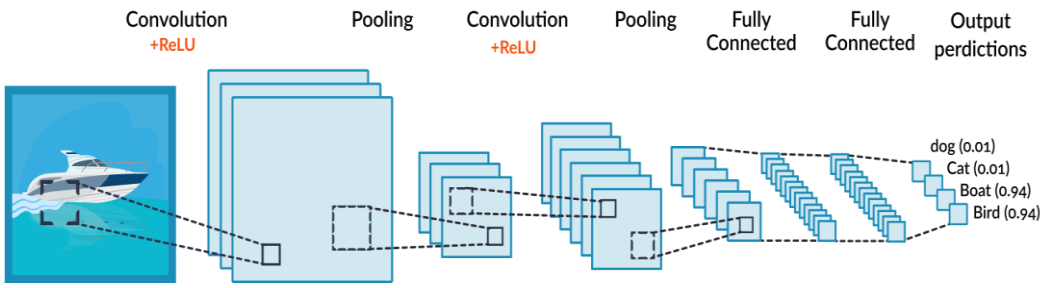
FFNNs yani İleri Beslemeli Yapay Sinir Ağları, geliştirilen yapay sinir ağlarının ilk ve en basit türüdür. Bu ağda, bilgi, girdi düğümlerinden ve gizli düğümler aracılığıyla, sadece bir yönde ileri çıkış düğümlerine doğru gider. Ağda döngü yoktur. Sadece ileri beslemeli güncelleme yapılır [25]. FNNs yöntemi şekil 3.15'de gösterildiği gibi uygulanır.



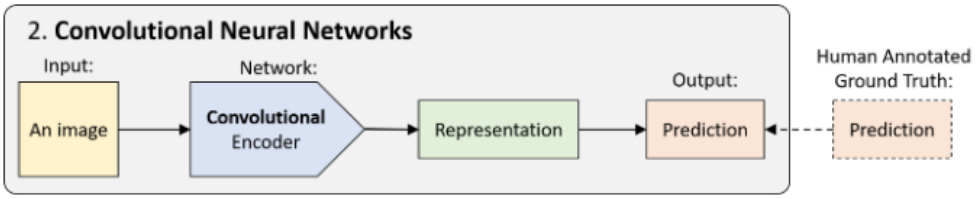
Şekil 3.15 : FFNNS Mimarisi [25].

3.1.1.2.2 Convolutional neural network

Sinir ağlarında, Evrişimli sinir ağı (ConvNets veya CNN), görüntü tanıma, görüntü sınıflandırmalarını yapmak için ana kategorilerden biridir. Nesne tespitleri, tanıma yüzleri vb. CNN'lerin yaygın olarak kullanıldığı alanlardan bazılarıdır. FFNNs sistemlerin aksine dense layer kullanmaz. CNN sistemler de convolutional katman kullanılır. Bu yapı görüntü sınıflarındırlması, nesne algılama, ses işleme, video işleme gibi yapısında uzamsal değişmezlik olan veriler için kullanılmaktadır. CNN mimarisi genel olarak bir input, bir output, convolutional (evrişimlsel) katmanlar içeren hidden layerler, pooling layer, fully connected layer katmanın'dan oluşmaktadır. Şekil 3.16 ve şekil 3.17'de CNN mimarısı ve algoritması resmedilmiştir.



Şekil 3.16 : Örnek bir CNN Algoritması [26].



Şekil 3.17 : CNN Mimarisi [25].

Şimdi ise ilgili katmanların projemizde ki işlevleri hakkında bilgi edinmek gerekirse:

Input Layer : Bir Evrişimli Sinir Ağının girişi, çoğunlukla pikselli bir veri olan bir matristir. Farklı veri türlerine göre farklı CNN mimarileri oluşturulabilir. Bu nedenle, daha önce çıkarılmış MFCC'leri girdi olarak kullanacağız; konuşmayı çok verimli bir şekilde tanımlamanın yanı sıra, bir matris olduğu için CNN'lerin kullanımına çok uygun bir yapıya sahiptir.

Convolutional Layer : Bu katmanın amacı giriş datasında ki en çarpıcı özellikleri yakalamak için küçük filtreleri tüm input (girdi) boyunca kaydırmaktır. Giriş matrisimiz üzerinde filtre sol üst köşeden başlayarak bütün matrisi gezer. Bu işlem sonucunda çıktı matrisini elde ederiz. Bu matrise "Feature Map" denir. Örnek vermek gerekirse giriş datamız bir görüntü olsun. Bu filtre görüntü üzerinde hareket ederek ve basit matris çarpımını kullanarak, özelliklerimizi tespit ediyoruz. Filtrenin giriş datası üzerinde kaydırılması aşamasında bilinmesi gereken 2 önemli terim vardır; bunlar "stride" ve "padding" terimleridir.

Stride, Bu değer evrişim işlemi için ağırlık matrisi olan filtreyi görüntü üzerinde birer piksellik adımlarla ya da daha büyük adımlarla kaydıracağının bilgisini verir. Bu da doğrudan çıkış boyutunu etkileyen diğer bir parametredir. Giriş matrisi, filtre ve stride boyutuna göre çıkış matrisimizde farklılık göstermektedir. Bu hesaplama aşağıdaki formüllasyon ile gerçekleşir:

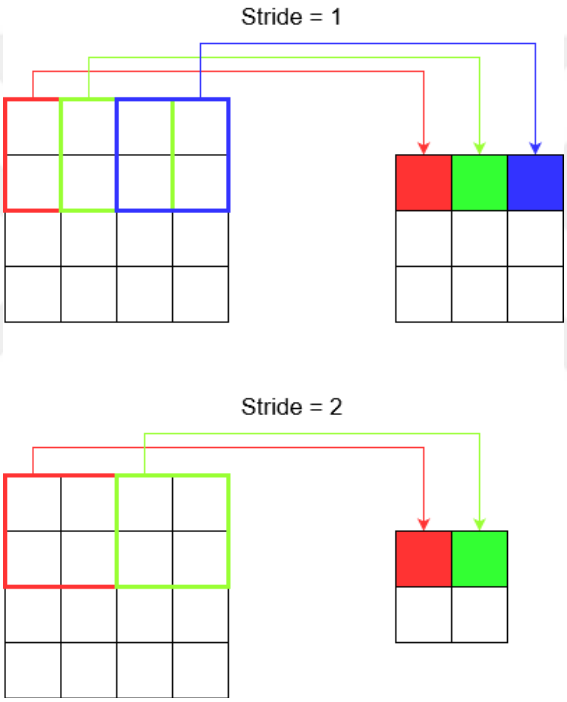
$$\text{Çıkış Matris Boyutu} = ((N - F) / S) + 1 \quad (3.19)$$

N: Giriş Matris Boyutu

F: Filtre Boyutu

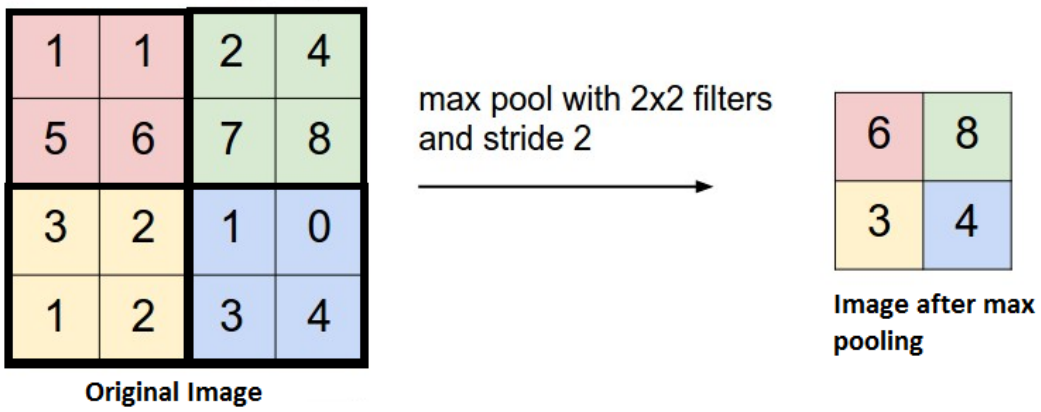
S: Stride yani adım boyutu

Padding, Şekil 3.18'de görüldüğü üzere giriş matrisimizi filtrelediğimiz zaman çıkış matrisinin boyutu küçülmektedir. İşte giriş matrisinin boyutunun azalmasını istemediğimiz durumlarda matrisin eksik boyutlarına değer atanmaktadır. Bu işleme de padding işlemi denir.



Şekil 3.18 : Stride Boyutu ve Çıkış Matrisi.

Pooling Layer : Bu katman genellikle CovNet'te birbirini izleyen kıvrımlı katmanlar arasına eklenir. Bu katmanın görevi, gösterimlerin yer değiştirmesini ve ağdaki parametreleri ve hesaplamaları azaltmaktır. Bu sayede ağda var olan bir uyumsuzluk kontrol edilmiş olur. Yani comvolutional layer'da hesaplanan çıktı pooling katmanında size reduction yöntemiyle daha küçük ve daha anlamlı bir dataya çevrilir. Aşağıda şekil 3.19'da görebilecğimiz üzere data üzerine uygulanmış 2x2'lik bir pooling layer katmanı datayı tarar ve her [2x2] frame için en yüksek değerli elemanı alır. Böylece değersiz datalar atılarak daha verimli ve daha maliyetsiz bir eğitim gerçekleşmiş olur.

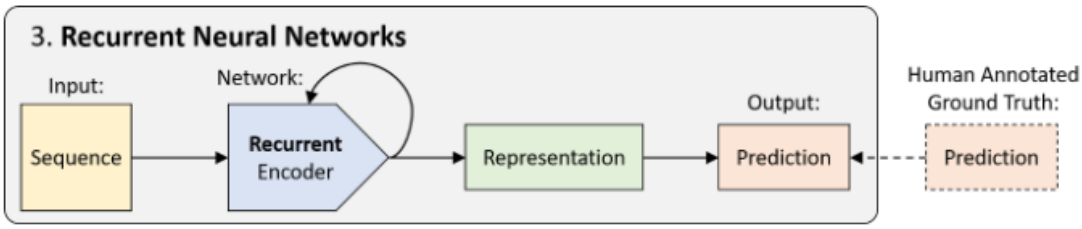


Şekil 3.19 : Pooling İşlemi.

Fully Connected (FC): Bir sinir ağındaki Fully Connected yani Tam Bağlantılı Katmanlar, bir katmandaki tüm girdilerin bir sonraki katmanın her etkinleştirme birimine bağlandığı katmanlara denir. En popüler makine öğrenme modellerinde, son birkaç katman, son çıktıyı oluşturmak için önceki katmanlar tarafından çıkarılan verileri derleyen tam bağlantılı katmanlardır. Bu katman Convolutional Layer'dan (Konvolüsyon Katmanı) sonra en çok zaman alan ikinci katmandır.

3.1.1.2.3 Recurrent neural networks (RNNs)

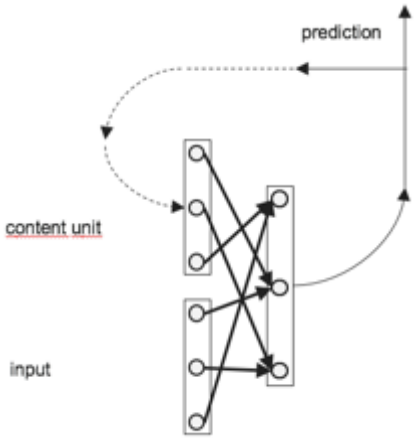
İlk başta şimdiye kadar baktığımız sinir ağlarının kalıcılığı yoktur, yani her iterasyon arasında öğrenilen tüm bilgiler silinir. Fakat bu bilgilerin önemli olduğu yerler vardır. Örnek vermek gerekirse yazı, konuşma, zamana bağlı çeşitli sensör veya istatistik amaçlı kullanılan veriler gibi belirli sırayla gelen ve sıra bilgisinin önemli olduğu konular için Recurrent Neural Networks yani Tekrarlayan Sinir Ağları yöntemi bir çözümdür. Şekil 3.20'de RNN mimasırı resmedilmiştir.



Şekil 3.20 : RNN Mimarisi [25].

Bu yapılar, outputlarını sonraki işlemde input olarak kullandıkları için feedforward yapılardan ayrılmış olurlar. Recurrent ağların bir belleğe sahiptirler. Bir ağa memory eklemenin sebebi ise, belli bir düzende gelen input setinin, çıktı için bir anlamı olmasıdır. Bu çeşit data setleri için feedforward ağlar yeterli olmaz.

Tam bu noktada RNN'ler devreye girer. Yazı, konuşma, zamana bağlı çeşitli sensör veya istatiksel veriler gibi belli bir sıra ile gelen verilerin yapısını anlamada recurrent ağlar kullanılır. Şekil 3.21'de RNN döngüsü yer almaktadır.



Şekil 3.21 : RNN Döngüsü [27].

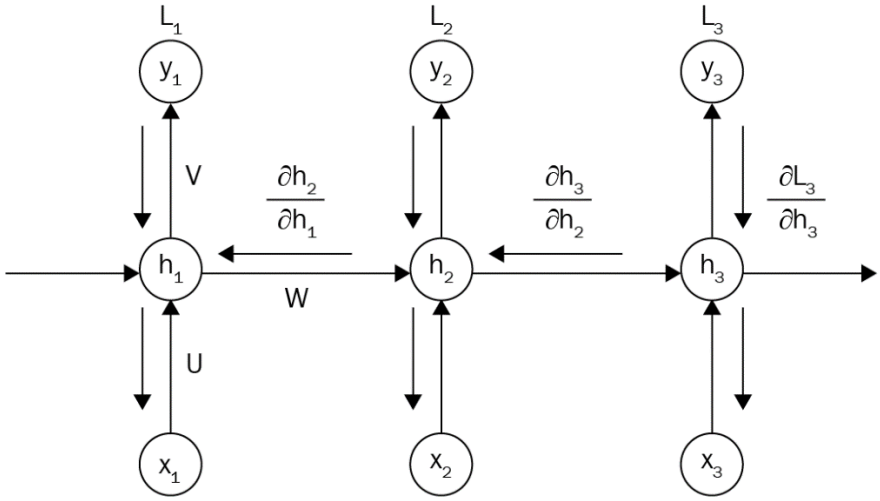
Farklı zamanlarda belleğe alınan veriler arasında korelasyon bulunuyorsa buna "long term" bağımlılık denir. İşte, RNN sistemler kullanıcısına bu long-term bağımlılıkların arasındaki ilişkiyi hesaplayabilen bir ağ yapısı sunarlar. İnsanların davranışlarında, konuşma ve düşüncelerinde de bu yapıdaki gibi önceden memory'de olan bilgi tıpkı bir hidden layer'da yeni veri ile döngüye girerek işlenir. Bu işlemi yaparken kullanılan matemetiksel formül aşağıdaki gibidir:

$$h_t = \Phi(\mathrm{W}x_t + Uh_{t-1}) \tag{3.20}$$

ht, t anındaki hidden layer'ın sonucunu gösteririr bize. xt inputu W ağırlığı ile çarpılır. Daha sonra t-1 anında, content unit'te tutulan h(t-1) değeri U ağırlığı ile çarpılır ve Wxt ile toplanır. W ve U değerleri girdi ile katman arasındaki ağırlıklardır. Burada ağırlık matrisi önceki ve şimdiki verinin hangisinin sonuca etkisi daha çok veya az ise ona göre değerler alır. Bu işlemler sonucunda oluşan hata hesaplanır ve backpropagation ile yeni ağırlık değerleri tekrar düzenlenir. Backprop işlemi hata yeterince minimize edilene kadar devam eder. Wxt + Uh(t-1) toplamı ise sigmoid, tanh gibi aktivasyon fonksiyonlarına sokulmaktadır. Böylece çok büyük veya çok küçük değerler mantıklı bir aralığa alınır.

Recurrent ağların sıralı inputları doğru bir şekilde sınıflandırmak için kullanılan geri yayılım mekanizmasına Backpropagation Trough Time (BPTT) yani Zaman İçinde Geri Yayılım Mekanizması kullanılır. Bu işlemi gerçekleştirebilmemiz için hatanın backpropunu ve gradient descentini kullanmamız gerekmektedir.

Backprob, çıkışındaki hatayı geriye doğru hatanın ağırlığına atayarak, ileri besleme ağında geri yayılma gerçekleştirilebilir. Bu türev kullanılarak, öğrenme katsayıları gradyanlı bir şekilde düzenlenir ve ağırlıklar da hataları azaltmak için düzenlenir. Şekil 3.22'de geriye yayılım algoritmasının şematiği görülmektedir.



Şekil 3.22 : Backpropagation işlem döngüsü [27].

Matematiksel denklem modeli aşağıdaki gibi verilmiştir:

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial y_3} * \frac{\partial y_3}{\partial h_3} * \frac{\partial h_3}{\partial W} \tag{3.21}$$

$$\frac{\partial L_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial L_3}{\partial y_3} * \frac{\partial y_3}{\partial h_k} * \frac{\partial h_k}{\partial W} \tag{3.22}$$

Burada h3'ün açılımında h2'ye bir bağımlılık bulunmaktadır. Bu durumu çözmek amacıyla zincir kuralını kullanarak h3'ün h2'ye türevini de ekleyerek sonucu elde edebilriz. Böylece formüldeki gibi gradient zamana bağlı şekilde dağıtılmış olmaktadır.

Backpropagation esnasında RNN sistemler gradyan kaybolması problemini ile karşı karşıya kalırlar. Bu problemin temel sebebi, aktivasyon fonksiyonlarının ürettiği sonuçlarının [-1, +1] aralığında oldukları sürece bacpropagation algortiması kullanımında, çarpılarak sıfıra yakınsar. İşte bu durumun önüne geçmek için W değerini uygun seçmek bu etkiyi azaltacaktır. Başka bir çözüm, tanh veya sigmoid aktivasyon fonksiyonları yerine ReLU kullanmaktır. ReLU fonksiyonunun türevi 0 veya 1 olmaktadır.

Bu nedenle, böyle bir sorun dahil edilmeyecektir. Başka bir yöntem de bu problemi çözmeyi amaçlayan LSTM yöntemidir. RNN sisteminin avantajları ve dezavantajları Çizelge 3.2'de gösterilmektedir;

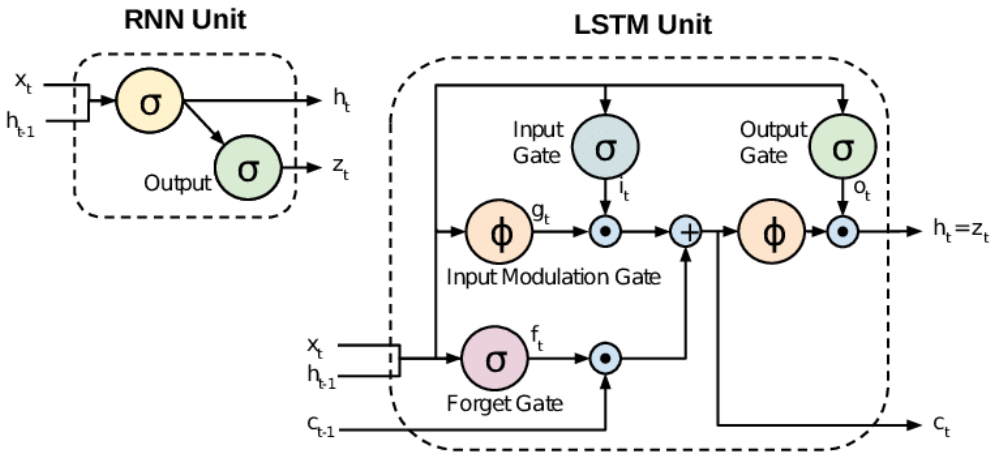
Çizelge 3.2 : RNN Mimarisi Avantaj ve Dezavantajları.

RNN Mimarisi Avantajları	**RNN Mimarisi Dezavantajları**
Bir dizinin her öğesi için aynı görevi önceki çıktılara bağlı olarak yerine getirir.	Autoencoder
RNN sistemler konvolüsyon katmanları ile birlikte kullanılarak etiketlenmiş görüntüler için tanımlayıcı üreten bir model olarak kullanılabilirler.	RNN sistemlerin eğitimi özellikle gömülü sistemler için flash, ram maliyetini arttırmaktadır.
	Aktivasyon fonksiyonu olarak tanh veya relu kullanılırsa çok uzun sekansları işleyemez.

Long Short Term Memory (LSTM): Genellikle “LSTM'ler” olarak adlandırılan Uzun Kısa Süreli Bellek ağları, uzun süreli bağımlılıkları öğrenebilen özel bir RNN türüdür. Hochreiter & Schmidhuber (1997) tarafından geliştirilmiştir. Geliştirilme amacı ise backpropagation esnasında farklı zaman ve katmanlardan gelen hata değerlerinin yok olmasını engellemektir. LSTM sistemlerde birbirine bağlı 4 kapı bulunur. Bunlar, Input Modulation Gate, Input Gate, Forget Gate, Output Gate’dir. Şimdi bu sistemin özellikleri üzerinde durmak gerekirse;

- Bilgiler depolanabilir, hücreye yazılabilir ve okunabilir.
- Hücre neyin saklanacağına ve ne zaman okunmasına, yazılmasına veya silinmesine izin verileceğine karar verir.

Bu kapılar bir ağ yapısına ve aktivasyon fonksiyonuna sahiptir.Aynı nöronlarda olduğu gibi gelen bilgiyi ağırlığına göre geçirir veya durdurur. Bu ağırlıklar ise RNN ağın öğrenmesi sırasında hesaplanmaktadır. Bu yapı ile hücre, datayı alacak mı, bırakacak mı yoksa silecek mi öğrenmiş olur. Şekil 3.23’de detaylar görülmektedir.



Şekil 3.23 : RNN Mimarisi ve LSTM Kapıları [27].

3.2 Kullanılan Yapay Zekâ Kütüphaneleri

Başlık 3.1'de yapay öğrenme metodları, aktivasyon fonksiyonları, algoritmaları ve matematiksel modelleri üzerinde durulmuştu. Bu başlıkta ise bu sistemlerin hangi programlama dilleri ile tasarlandığı ve kullanılan kütüphanler üzerinde durulacaktır. [28]

Çizelge 3. 3 : Yapay Zekâ Kütüphanleri.

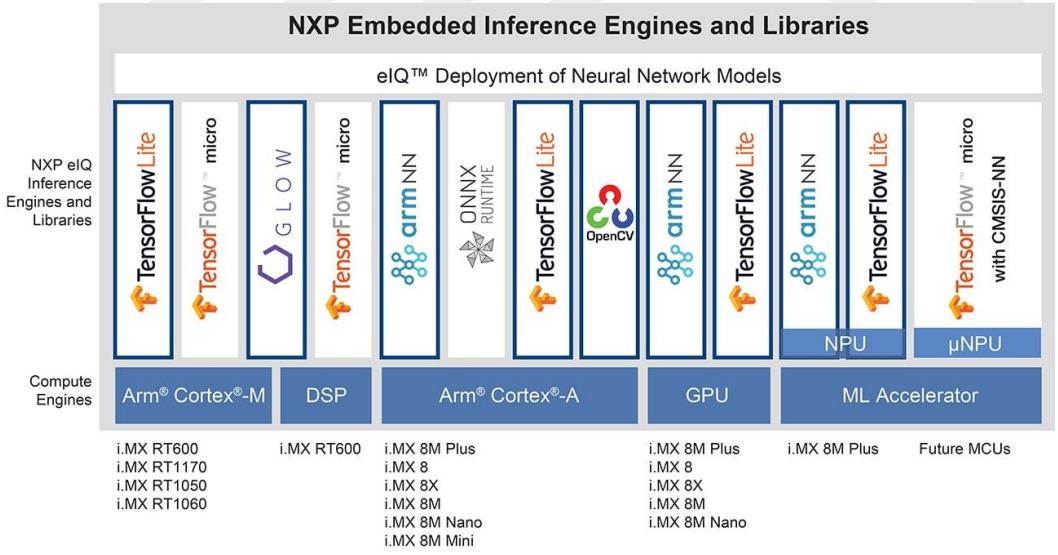
Kütüphane	**Programlama Dili**	**Geliştirici**
TensorFlow	Python	Google
Torch	Lua	Ronan Collabert
Digits	C++	Nvidia Corporation
Theano	Python	MILA LAB
Caffe	Python	BVLC
KNET	Julia	Deniz Yuret
DeepLearning4j	Julia	Adam Gibson

Yapılan tez kapsamındaki TensorFlow kütüphanesi; esnek yapısı, tek bir API kullanarak platformdan bağımsız hesaplamalar, bir veya daha fazla CPU ve GPU kullanan uygulamaları destekleyen ve Python kullanarak geliştirmeye dayanan bu çerçeve yapısı, birden fazla dili destekleyecek.

Örneğin C ++, Java, C #, Javascript ve R ve bugünün Python'u. Kağıtta kullanılan gömülü işlemci mimarisiyle uyumlu olduğu için tercih edilir. [29]

3.2.1 Yapay zekâ kütüphanesi olarak: TensorFlow

TensorFlow temel olarak Google tarafından geliştirilmiş Python programlama altyapısı kullanan açık kaynak kodlu bir derin öğrenme kütüphanesidir. Aynı zamanda R, C++, Java gibi dillerdeki TensorFlow uygulamaları geliştirmek için dile ait olan kütüphaneyi kullanabilmek için programcıya olanak sağlamaktadır. Aynı zamanda TensorFlow Lite versiyonu mikrodenetleyiciler ve sadece kilobayt belleğe sahip cihazların makine öğrenme modelleri çalıştırmak için tasarlanmış bir modeldir. Bu model doğrultusunda PC ortamında tasarlanan derin öğrenme modelini gömülü sistem mimarisine aktarılması mümkün kılınmıştır. Bu yapı NXP Semiconductors firması tarafından tasarlanan eIQ platformu kapsamında gerçekleştirilmiştir. İlk başta eIQ yapısının ne olduğu ve işlevi hakkında bilgi vermek gerekirse NXP eIQ (“edge intelligence”) yapısı Machine Learning yazılım ortamı, gömülü sistemlerde NN (Neural Network) yapay zekâ (AI) modelleri ile sonuç çıkarmak için NXP mikroişlemcileri ve mikrodenetleyicilere için temel bileşenler sağlar. Google TensorFlow Lite, Arm NN, Arm CMSIS-NN ve OpenCV'yi destekleyen çıkarım motorları, NN derleyicileri, kütüphaneler ve donanım soyutlama katmanlarını içerir. [30] Şekil 3.24’de kütüphaneler görülmektedir.



Şekil 3.24 : Makine Öğrenmesi: NXP eIQ [31].

Öncelikle TF Lite, FlatBuffers'a dayalı bir model dosya formatı tanımlar. TF’nin protokol arabelleklerinden farklı olarak, FlatBuffers'ın önbellek hatlarının daha iyi kullanılmasına izin veren daha küçük boyutta bir bellek alanı mevcuttur ve bu da NXP cihazlarında daha hızlı yürütmeye neden olur.

TF Lite, TF sinir ağı operasyonlarının bir alt kümesini desteklese de, hala geniş bir yelpazede desteklenen işlemleri sürdürür, böylece CMSIS-NN'ye kıyasla daha gelişmiş modellerin kullanılmasına izin verir. Aynı zamanda tekrarlayan sinir ağlarını (RNN) ve daha da karmaşık uzun kısa süreli bellek (LSTM) hücrelerini destekler. [32]

eIQ ML (Machine Learning) yazılımı, i.MX 8 uygulama işlemcisi ve i.MX RT cross over işlemci ailelerinde TensorFlow Lite'ı desteklemektedir ve sırasıyla Yocto ve MCUXpresso ortamlarında kullanılabilir. TensorFlow Lite, kullanıcıların TensorFlow modellerini daha hızlı çıkarımlar gerçekleştirmeleri ve daha az bellek gerektirmeleri için dönüştürmelerine ve dağıtmalarına olanak tanıyan ve kaynak kısıtlı, düşük güçlü cihazlar için ideal hale getirmesini sağlayan imkanlar sağlar.

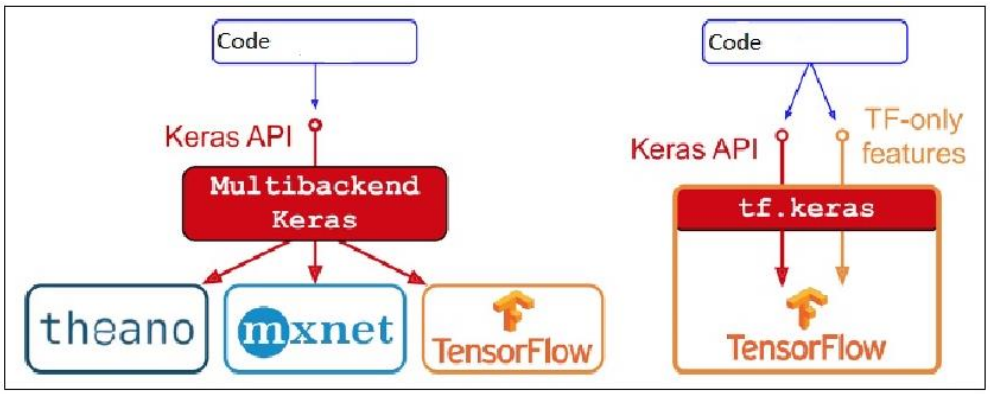
TensorFlow kütüphanesinin makine öğrenimi için end to end açık kaynaklı bir platform olduğunu belirtmiştik. Üst düzey API'lerle iş akışları sağlayan kapsamlı ve esnek bir araç, kitaplık olduğu için de tez kapsamında ki tercihimiz olduğunu bilmekteyiz. Şimdi ise TensorFlow ve Theano gibi platformların üstünde çalışan ve tez kapsamında kullanılan üst düzey bir yapay sinir ağları kütüphanesi olan Keras Kütüphanesi hakkında bilgi verilecektir. Keras'ı derin öğrenme alanında kullanmak, CPU ve GPU'da sorunsuz bir şekilde çalışmanın yanı sıra kolay ve hızlı prototip oluşturmamıza izin vermektedir. Bu çerçeve hata ayıklaması kolay olan ve genişletilebilirliği kolaylaştıran Python koduyla yazılmıştır. Keras'ın ana avantajları aşağıdaki gibi detaylıca verilmeye çalışılmıştır;

Kullanıcı Dostu Tasarım: Keras, genel kullanım durumları için optimize edilmiş, kullanıcı hataları için net ve işlem yapılabilir ayrıca geri bildirim sağlayan basit, tutarlı bir arayüze sahiptir.

Modüler Yapı: Keras modelleri, yapılandırılabilir yapı taşlarını birkaç kısıtlama ile birbirine bağlayarak yapılır.

Genişletilmesi Kolay: Keras'ın yardımıyla yeni fikirler ve araştırmalar için kolayca özel yapı taşları yazabiliriz.

Kullanımı Kolay: Keras, yaygın kullanım durumları için gereken kullanıcı eylemlerinin sayısını en aza indirmeye yardımcı olan tutarlı ve basit API'ler sunar, ayrıca kullanıcı hatası üzerine net ve işlem yapılabilir geri bildirim sağlar.



Şekil 3.25 : Keras uygulanma metodları [33].

3.2.1.1 Keras kütüphanesinin TensorFlow ile kullanılması

tf.keras komutu, TensorFlow'un Keras API spesifikasyonunu uygulamasını sağlar. Bu, katman tf.data TensorFlow'a özgü işlevler için birinci sınıf destek içeren modeller oluşturmak ve eğitmek için üst düzey bir API'dır. tf.keras esneklik ve performanstan ödün vermeden TensorFlow'un kullanımını kolaylaştırır. tf.keras komutu şekil 3.26'da gösterildiği koda import edilir.

```
import tensorflow as tf

from tensorflow import keras
```

Şekil 3.26 : Keras kütüphanesinin TensorFlow implementasyonu.

Keras TensorFlow'un ölçeklenebilirlik ve platformlar arası özelliklerinden tam olarak yararlanması için imkân verir: Keras'ı TPU'da veya büyük GPU kümelerinde çalıştırabilir ve Keras modellerini tarayıcıda veya bir mobil cihazda çalıştırmak için dışa aktarabiliriz.

Keras / TensorFlow uyumluluğu aşağıda verilen programlar ile uygundur:

Python 3.5–3.7

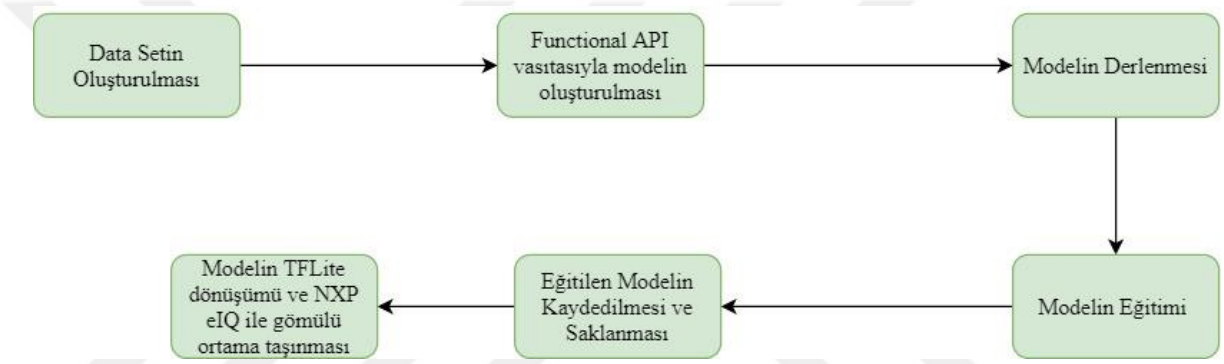
Ubuntu 16.04 veya sonrası

Windows 7 veya sonrası

macOS 10.12.6 (Sierra) veya sonrası

Keras Kütüphanesi ve TensorFlow ile Derin Öğrenme Algoritmasının Geliştirilmesi

Tensorflow üzerinde çalıştırılan Keras kütüphanesi sayesinde YSA modelinin oluşturulmasından, elde edilen modelin kaydedilmesine kadar olan süreci yönetebilmekteyiz. Bu başlık altında bu sürecin nasıl yürütüldüğü genel bir biçimde ele alınacaktır. Şekil 3.27'de verilen yol haritası izlenecek ve CNN ve LSTM modeller üzerinde Keras kütüphanesi çalışan Tensorflow yardımıyla elde edilecektir. Model, Google tarafından geliştirilen CoLab ortamı kullanılarak tasarlanacaktır. Şekil 3.27'de model tasarım aşamaları yer almaktadır.



Şekil 3.27 : Model Tasarım Aşamalar.

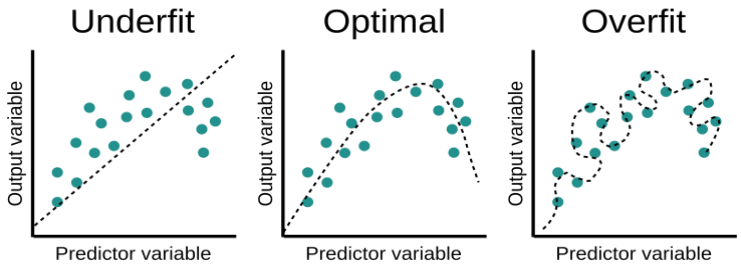
Data Set Oluşturulması: Data Set yani "Veri Seti" kavramı bir veri topluluğununu ifade etmektedir. Başka bir deyişle, bir veri kümesi, bir tablonun her sütununun belirli bir değişkeni temsil ettiği ve her satırın söz konusu veri kümesinin belirli bir üyesine karşılık geldiği tek bir veritabanı tablosunun veya tek bir istatistiksel veri matrisinin içeriğine karşılık gelir. YSA projelerinde bir eğitim veri setine ihtiyacımız vardır. Kullanma amacımız ise modeli çeşitli eylemler gerçekleştirmek üzere eğitmekdir. Yani ortamdan topladığımız gerçek verileri daha öncede öğrendiğimiz gibi "supervised learning" tekniğini kullanarak geliştireceğimiz modeli eğitmek ve daha sonra ortamdan alacağımız veriyi bu eğitim doğrultusunda sınıflandırmak için kullanırız. Bu noktada data set 2'ye ayrılmaktadır. Bunlar training dataset ve test dataset'tir. Bu dataset kümesinin kullanım yerlerini detaylıca ele alalım. Training dataset, sinir ağları gibi kavramların nasıl uygulanacağını, sonuçların öğrenilmesi ve üretilmesi için bir algoritma eğitmek için kullanılan bir settir. Hem giriş verilerini hem de beklenen çıktıyı içerir. Eğitim setleri toplam verilerin en az %60 oranında

çoğunlunu oluşturur. Testlerde, modeller, ağırlıkların ayarlanması olarak bilinen bir işlemdeki parametrelere uyar.

Test data set'ine geldiğimizde ise algoritmanın egzersiz veri seti ile ne kadar iyi eğitildiğini değerlendirmek için kullanılır. Yapay zekâ projelerinde, test aşamasında ayarlanan eğitim verilerini kullanamayız çünkü algoritma, hedefimiz olmayan beklenen çıktıyı önceden zaten bilecektir. Test setleri verilerin en fazla %20'sini temsil eder. Test setinin, genellikle insan doğrulaması ile doğrulanmış doğru çıkışlarla birlikte gruplandırılmış girdi verileri olması sağlanmaktadır. Data Set'in doğru tayini bir YSA modeli geliştirmede en kritik rolü üstlenmektedir. Şimdi verilecek olan iki kavram bunun öneminin çok daha iyi anlaşılmasını sağlayacaktır. Bu kavramlar, Overfitting ve Underfitting kavramlarıdır. Overfitting kavramı modelinizin verilen veri setini öğrenmek yerine ezberlemesine verilen isimdir. Bu genellikle model çok karmaşık olduğunda (yani gözlem sayısına kıyasla çok fazla özellik veya değişken varsa) gerçekleşir. Bu model eğitim verilerinde çok yüksek tahmin doğruluğuna sahip olacaktır, ancak eğitimsiz veya yeni verilerde muhtemelen çok doğru tahminleme yapamayacaktır. Bu tip durumlarda yüksek varyans, düşük bias durumu görülmektedir. Bu sorun modelin genelleştirme yapamamasından kaynaklanmaktadır. Bu tip modeller verilerdeki değişkenler arasındaki gerçek ilişkiler yerine eğitim verilerindeki "gürültüyü" öğrenirler.

Underfitting kavramı ise örüntüyü ezberlemek yerini anlamaya çalışmak, fakat yeterince anlamamaktır. Size verilen örüntünün kuralını değil, istenilen dışında çıktılar veren başka bir kuralı kullanması anlamına gelir. Underfitting sorunu olan modellerde hem eğitim hem de test veri setinde hata oranı yüksektir. Düşük varyans ve yüksek bias'a sahiptir. Bu noktada bias (yanlılık) ve varyans kavramlarını açacak olursak;

Sapma, model hatasının derecesini ölçebilir. Hataları genellemenin bu kısmı yanlış varsayımlara dayanmaktadır. Örneğin, veriler kuadratik (kuadratik polinom) olduğunda verilerin doğrusal olduğunu varsayalım. Sapmalar, modelimizin bu soruna bir çözüm içermediğini göstermektedir. Bu durumda, modelimiz zayıftır ve yetersiz uyum oluşacaktır. Yüksek bias değerine sahip modellerin sahip olmadığımız eğitim verilerini anlama olasılığı daha yüksektir. Varyans ölçüm modelinin tahmin verilerinin gerçek verilere nasıl dağıldığı. Genelleme hatasının bu bölümü, modelin eğitim verilerindeki düşük değişikliklere karşı çok hassas olmasından kaynaklanmaktadır. Varyans yüksekse, modelimiz çok geneldir; bu da overfitting durumuna yol açar.



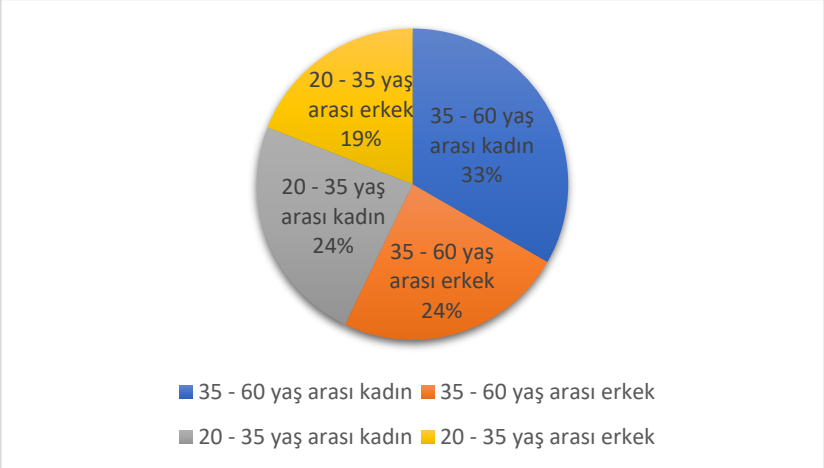
Şekil 3.28 : Data Set'e bağlı model eğtimi çıktıları [34].

Bu kavramlar ışığında tez kapsamında kullanılan dataset Google AI blog tarafından 65000 kişinin katkısıyla toplanmış ve 1 saniye uzunluğundaki verilerden yararlanılmıştır. Bu dataset vasıtasıyla tez kapsamında kullanılan etiket kelimeleri ise Şekil 3.29'da verilmiştir;

```
const std::string labels[] = {"bed", "eight", "five", "off",
      "on", "right", "sheila", "silence", "unknown", "zero"};
```

Şekil 3.29 : Tez kapsamında kullanılan etiket kelimeler.

Ses tanıma çalışmasında eğitim ve test datalarının çeşitlilik göstermesi çok önemlidir. Bunun sebebi ise "Ses İşleme" başlığı altında da belirtildği gibi sesin dinamik yapısının çok çeşitli bir karakteristiğe sahip olmasıdır. Yani aynı kelimenin söylendiği zaman elde ettiğimiz karaskteristik özellikleri kişiden kişiye çok büyük farklılıklar göstereceği gibi aynı kişinin farklı söylemlerinde de değişiklikler göstermektedir. Bundan dolayı kullanılan veri seti yaş ve cinsiyet bakımında çeşitli tutulmuş ve bir kişiden yaklaşık olarak 70 – 80 arası ses kaydı örneği alınmıştır. Aynı zamanda cinsiyet – yaş dağılımını içeren grafik 3.30'de görülebilmektedir.



Şekil 3.30: Data set yaş ve cinsiyet dağılımı.

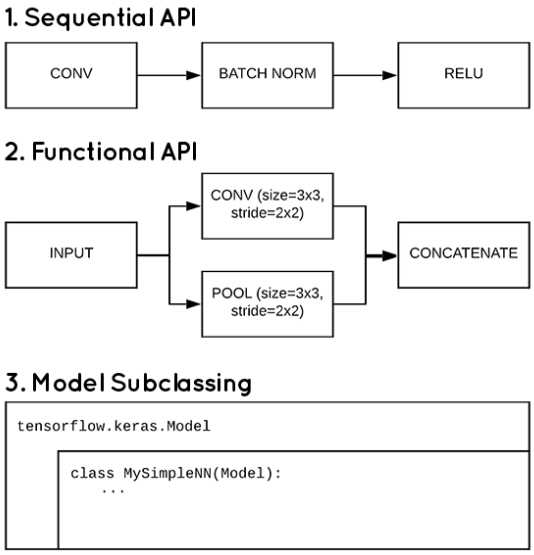
Functional API vasıtasıyla model oluşturulması: Konuyu en geniş kapsamda ele alacak olursak Keras kütüphanesi vasıtasıyla model yaratmak için 3 yol mevcuttur. Bunlar sırasıyla, Sequential Model, Functional API ve Model Subclassing metodlarıdır.

Sequential Model, basit bir katman listesi ancak tek girişli, tek çıkışlı katman yığınlarıyla sınırlı Sıralı model tasarımları için kullanılır.

Functional Model, keyfi model mimarilerini destekleyen, kullanımı kolay, tam özellikli bir API'dir. Çoğu insan ve çoğu kullanım durumunda, kullanmanız gereken şey budur. Bu Keras "endüstri gücü" modelidir.

Model Subclassing, tamamen özelleştirilebilir ve modelin kendi özel ileri geçiş uygulamasını sağlar.

Bununla birlikte, bu esneklik ve özelleştirmenin maliyet modeli alt sınıflamasında kullanılması Sequential API veya Functional API'den çok daha zordur. Egzotik mimariler veya özel katman model uygulamaları, özellikle araştırmacılar tarafından kullanılanlar, standart Sequential veya Functional API'leri kullanarak uygulanması imkânsız olmasa bile, son derece zor olabilir. Şekil 3.31 de keras modelleri belirtilmiştir.



Şekil 3.31: Keras Modelleri [35].

Amaçlarımıza göre bir YSA modelinin Keras kütüphanesi kullanılarak hangi metodlar ile kurulacağı konusu üzerinde duruldu. Şimdi ise üzerinde durulacak olan nokta ise tasarlanacak olan modelde kullanılacak olan Functional API metodunun geliştirilmesidir. Keras'ın Functional API'sinin kullanımı genel olarak kolaydır ve

Keras derin öğrenme kütüphanesini kullanan çoğu derin öğrenme uygulayıcısı tarafından tercih edilir.

Functional API'yı kullanarak aşağıda verilen özellikleri kolaylıkla yapabilmekteyiz:

- Karmaşık modellerin oluşturulması mümkündür.
- Birden çok giriş ve birden çok çıkışlı model tasarımı sağlar.
- Tasarlanan Mimarideki dalları kolayca tanımlayabiliriz.

Functional API metodu 3 önemli modül içermektedir. Bunlar conv_module, inception_module, downsample_module'dür. Detaylı incelemek gerekirse;

Conv_module: Bir giriş biriminde konvolüsyon gerçekleştirir, toplu normalleştirmeyi kullanır ve ardından bir ReLU aktivasyonu uygular. Bu modülü basitlikten ve yeniden kullanılabilir hale getirmek için tanımlarız, mümkün olduğunca az kod satırı kullanarak, mimarimizi içine kolayca "convolutional block" uygulayabilmemizi sağlayarak, uygulamamızı düzenli, düzenli ve hata ayıklamayı kolaylaştırırız.

Inseption_module: İki conv_module bloğunu başlatır.

Birinci conv bloğu 1 × 1 evrişim uygularken, ikinci blok "aynı" dolgu ile 3 × 3 evrişim gerçekleştirir ve 1 × 1 ve 3 × 3 evrişimli çıkış hacmi boyutlarının aynı olmasını sağlar. Çıkış hacimleri daha sonra kanal boyutu boyunca birleştirilir.

Downsample_module: Bu modül, bir giriş hacminin boyutunu azaltmaktan sorumludur. İnception_module'e benzer şekilde burada iki dal kullanılır. İlk dal, 3 × 3 konveksiyon gerçekleştirir, ancak 2 × 2 adım ve dolgu ile hacim boyutunu azaltır. İkinci dal, 2 × 2 adım ile maksimum 3 × 3 havuzlama uygular. Her iki dal için çıkış hacmi boyutları aynıdır, böylece kanal ekseni boyunca birlikte birleştirilebilirler.

Aslında bu modüllerin her birini legolar olarak düşünüyoruz. Her bir lego tipini uyguluyoruz ve daha sonra model mimarimizi tanımlamak için belirli bir şekilde istifliyoruz. Legolar neredeyse sonsuz sayıda olasılıkla organize edilebilir ve birbirine uyarlanabilir; ancak, form işlevi tanımladığından, dikkat etmeli ve bu legoların nasıl birbirine uyması gerektiğini düşünmemiz gerekmektedir.

Modelin Derlenmesi: Tasarlanan modelin derlenmesi aşamasında iki önemli parametre rol oynamaktadır; bunlar optimize edici (optimizer) ve kayıp (loss) değerleridir. Burada Optimize edici öğrenme oranını (learning rate) kontrol etmektedir.Tez kapsamında optimize edici olarak "adam" ı kullanacağız. Adam

genellikle birçok durumda kullanmak için iyi bir optimize edicidir. Adam (adaptive moment estimati), eğitim verilerine dayalı ağ ağırlıklarını yinelemeyi güncellemek için klasik stokastik gradyan iniş (gradient descent) yerine kullanılabilecek bir optimizasyon algoritmasıdır. Adam optimizer eğitim boyunca öğrenme hızını (lr) ayarlamaktadır. Tez kapsamında kayıp fonksiyonu olarak mean squared error (MSE), yani en küçük kareler yöntemi kullanılmaktadır. Bu method, tahmin edilen ve gerçek değerler arasındaki ortalama kare farkı alınarak hesaplanır. Regresyon problemleri için popüler bir kayıp fonksiyonudur. 0'a ne kadar yakın olursa, model o kadar iyi performans gösterir.

$$\text{MSE} = \frac{1}{N} * \sum_{i=1}^{N} (f_i - y_i)^2 \quad (3.23)$$

N: Veri miktarı

f_i : Modelden elde edilen geri dönüş değeri

y_i : Gerçek değer

```
Epoch 40/40
271/271 [==============================] - 2s 7ms/step - loss: 0.2827 - accuracy: 0.9403 - val_loss: 0.4992 - val_accuracy: 0.8803
```

Şekil 3.32 : CNN modelden elde edilmiş başarım oranı

Modelin Eğitimi: Modelin eğitilmesi aşamasında fit() fonksiyonunu, 5 parametre ile birlikte kullanacağız. Bu parametreler training data, target data, batch_size ve epoch ve callback sayısı'dır. Bu parametreleri fonksiyon içerisine yerleştirirsek model.fit(train_X, train_Y, batch_size, epoch) yapısını elde ederiz. Bu noktada parametrelerin açıklamasını yapmak gerekirse;

train_X: Eğitim Verisi

train_Y: Etiket Verisi

batch_size: Ağ üzerinden yayılacak örnek sayısını tanımlar. Örnek vermek gerekirse 1000 training sample olduğunu ve 100'e eşit bir batch_size ayarlamak istediğinizi varsayalım. Algoritma ilk 100 örneği (1'den 100'e kadar) eğitim veri kümesinden alır ve ağı eğitir. Ardından, ikinci 100 örneği alır (101'den 200'e kadar) ve ağı tekrar eğitir. Tüm örnekleri ağ üzerinden geçirene kadar bu işlemi yapmaya devam eder.

Epoch: Eğitim sırasında tüm eğitim verilerinin ağa gösterilme sayısıdır.

Eğitilen Modelin Kaydedilmesi ve Saklanması: İlk olarak bir Keras modelini oluşturan bileşenlere bakalım;

- Modelin hangi katmanları içerdiğini ve nasıl bağlandığını belirten bir mimari.
- Bir ağırlık seti ("modelin durumu").
- Optimize edici (model derlenerek tanımlanır).
- Bir dizi kayıp ve metrik (modeli derleyerek veya tf.keras.layers.Layer.add_loss veya tf.keras.layers.Layer.add_metric öğesini çağırarak tanımlanır).

İşte Keras API bu bileşenlerin hepsini bir kerede diske kaydedilmesini veya isteğe bağlı olarak bazılarının seçici olarak kaydedilmesini mümkün kılmaktadır;

- TensorFlow SavedModel formatında (veya eski Keras H5 formatında) her şeyi tek bir arşive kaydetme. Standart uygulama budur.
- Yalnızca mimariyi / yapılandırmayı, genellikle bir JSON dosyası olarak kaydetme.
- Yalnızca ağırlık değerlerini kaydetme. Bu genellikle modeli eğitirken kullanılır.

Tüm modeli diske kaydetmek için kullanabileceğiniz iki biçim vardır: TensorFlow SavedModel biçimi ve daha eski Keras H5 biçimi. Önerilen biçim SavedModel'dir. Model.save () kullandığınızda varsayılan değerdir. Şekil 3.33 ve 3.34'de keras modelleri görülmektedir.

H5 formatına şu şekilde geçebiliriz:

model.save(my_model.h5)

```
model = ...  # Get model (Sequential, Functional Model, or Model subclass)
model.save('path/to/location')
```

Şekil 3.33 : Keras Modelinin kayıt edilmesi.

```
from tensorflow import keras
model = keras.models.load_model('path/to/location')
```

Şekil 3.34 : Keras Modelin Yüklenmesi.

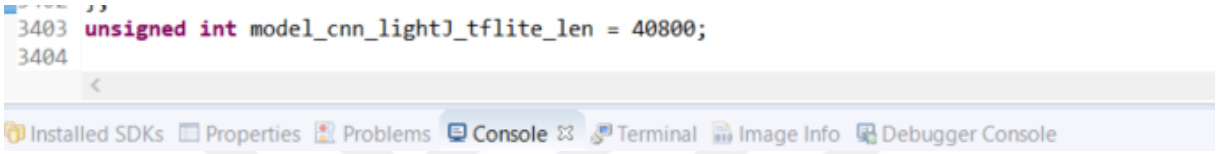
Modelin TFLite dönüşümü ve eIQ platformu ile gömülü sisteme taşınması: Bu aşamadan itibaren oluşturulan modelin TFLite dönüşümü ve gömülü işlemci mimarisine aktarılması kısmını ele alacağız. Bu aşamada elde edilen .h5 formatlı dosya C/C++ programlama diline uygun şekilde .h file formatında bir header file'a dönüştürülecek ve gömülü sistem için tasarlanan proje dosyasına eklenecektir. Şekil 3.35'de dönüşüm kodları yer almaktadır.

```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_keras_model(model_cnn
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
open("model_cnn_lightJ.tflite", "wb").write(tflite_model)
```

Şekil 3.35 : TensorFlow - TFLite Dönüşümü.

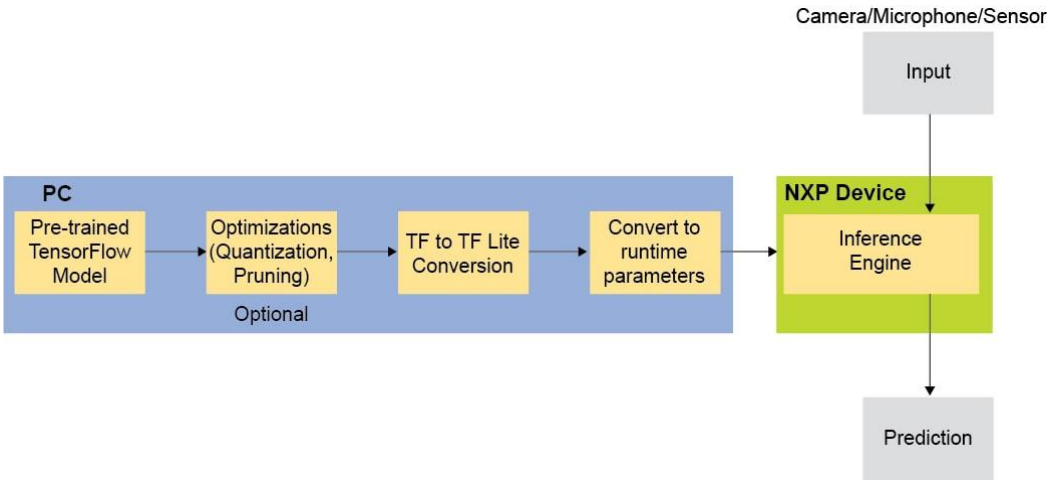
Bu komutun işlenmesi ile birlikte Colab üzerinden tasarladağımız modeli 40800 elemanlı model parametrelerini içeren bir C/C++ programlama dili için uyumlu bir header dosyası elde edilmektedir.



Şekil 3.36 : Model parametleri eleman sayısı.

3.2.1.2 TensorFlow çalışması sonucu elde edilen çıktılar

Keras Kütüphanesi kullanılarak Tensorflow üzerinde elde edilen modelin TFLite dönüşümü sağlanmış ve tezin amacı olan gömülü sistemler ile sesli komut algılama için önemli bir kilometre taşı daha aşılmıştır. Artık bir sesli komut setleriyle eğitilmiş bir model elde etmiş olduk. Bundan sonraki aşama ise Şekil 3.37'de de görüleceği üzere bir gömülü sistem üzerinde yerleştirilmiş mikrofonlar vasıtasıyla ses datası alıp eğitilmiş model doğrultusunda doğru etiketli kelimenin elde edilmesi olacaktır.



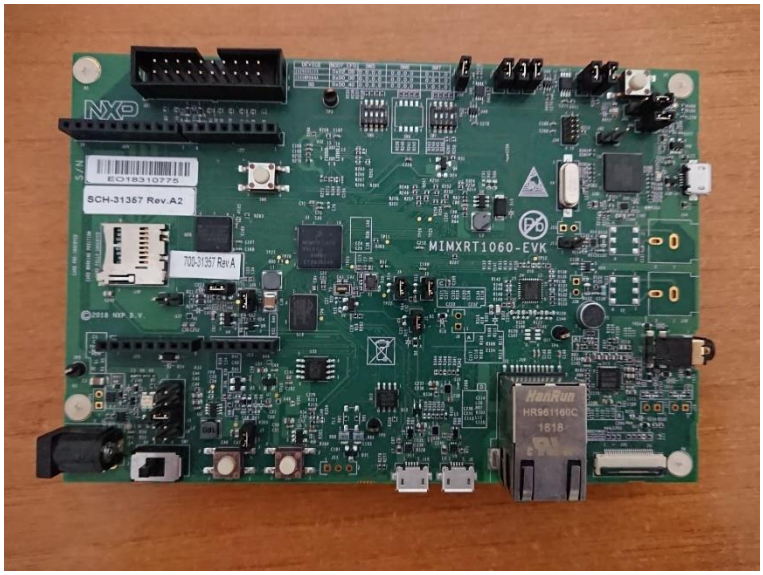
Şekil 3.37 : Gömülü Sistem Konuşma Tanıma Mimarisi Tasarımı [36].

4. GÖMÜLÜ SİSTEMLER MİMARİSİ TASARIMI

Bu bölüme kadar gelinen süreçte konuşma tanıma sistemleri, sinyal işleme ve bir derin öğrenme modelinin nasıl oluşturulacağı ve bu modelin nasıl gömülü sistem mimarisine entegre edileceği üzerinde duruldu. Şimdi ise kullanılan işlemci mimarisi, ram ve flash optimizasyonu, mikrofon üzerinden ses komutu alınması, alınan datanın işlemci üzerinde işlenerek tasararlanan model içerisinde doğru etiket sınıfına yerleştirilmesi aşaması ve C/C++ cross compile konuları üzerinde durulacaktır.

4.1 Tez Kapsamında Kullanılan İşlemci Mimarisi

Tez kapsamında 600 MHz işlemci hızına sahip ARM Cortex – M7 mimarisine sahip yüksek hızlı GPIO, CAN-FD ve senkron paralel NAND / NOR / PSRAM denetleyici gibi gerçek zamanlı uygulamalar için ideal ek özellikler sunan ve eIQ platformu ile derin öğrenme projelerini destekleyen NXP i.MX RT 1060 işlemcisi tercih edilmiştir [37]. Şekil 4.1'de tez kapsamında kullanılan geliştirme kartı verilmiştir.



Şekil 4.1 : MIMXRT 1060 Eval Board.

Tez kapsamında kullanılan geliştirme kartının ürün kodu aşağıdaki gibidir:

TYPE: MIMXRT1060 – EVK

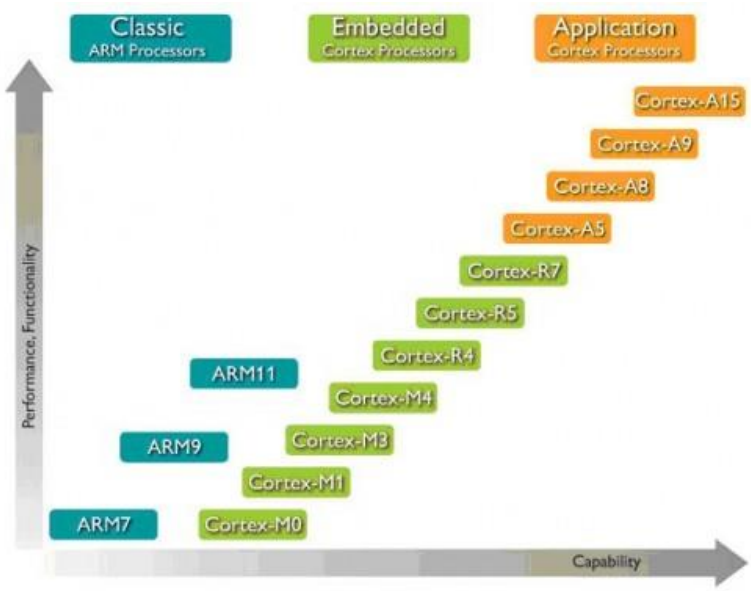
CODE NO: 9353 682 84598

Geliştirme kartına ait özellikler çizelge 4.1'de düzenlenmiştir;

Çizelge 4.1 : MIMXRT 1060 Eval Board Özellikleri.

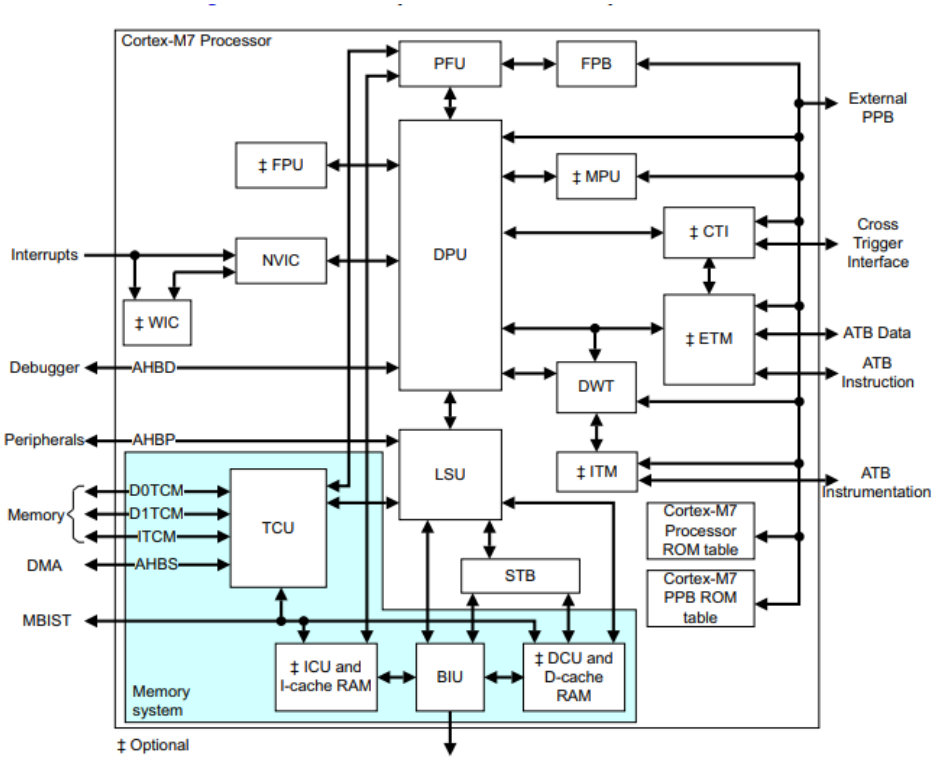
MIMXRT 1060 Eval Board
ARM Cortex M7 mimarisine sahip MIMXRT10 6 2DVL6A işlemcisi
3020 CoreMark / 1284 DMIPS @ 600 MHz
1 MB On – Chip SRAM, Boot ROM (128 KB)
3X I2S multi - channel Audio
High Speed gpio, CAN – FD ve senkron paralel NAND / NOR / PSRAM kontrolör
32 MB SDRAM (external)
8 MB QSPI Flash (external)
64 MB Hyper Flash
3.5 mm audio stereo kulaklık girişi
JTAG 20 – pin connector

ARM (Acorn RISC Machine), RISC tabanlı olarak geliştirilmiş işlemci mimari yapısına verilen isimdir. Şekil 4.2'de ARM Mimari yapısı verilmiştir.



Şekil 4.2 : ARM Mimari Yapısı.

Cortex M serisi ailesine ait olan M7 işlemcisi, düşük kesme gecikmesi (low input latency), düşük maliyetli hata ayıklama özelliklerine sahip ve mevcut ile geriye dönük uyumluluğu olan yüksek verimli, yüksek performanslı, gömülü bir işlemcidir. Şekil 4.3'de Cortex M7 işlemcisine ait fonksiyon diyagramı verilmiştir.



Şekil 4.3 : Cortex M7 Fonksiyon Diyagramı [38].

Floating Point (Kayan Nokta) İşlemleri

Birçok gömülü sistemler uygulamasında tamsayı aritmetiği kullanılarak uygulanabilse de, ses işleme gibi kayan nokta yani gerçek sayılarla başa çıkma yeteneğinin gerekli olduğu zamanlar vardır. Bu kısımda, kayan nokta işlemlerinin ayrıntılarına, kayan nokta (floating point) kullanılmalı ve kullanılmaması gerektiğine değineceğiz. ARM Cortex bir tamsayı (integer) işlemci olduğundan, ses işleme algortimalarında tüm kayan nokta işlemleri floating point aritmetiği kullanılarak gerçekleştirilmelidir. Bu noktada "floating point" ve "fixed point" kavramlarını detaylı açıklamak gerekirse; Günümüzde, çoğu gömülü sistem mimarisi 32 bit CPU'lar kullanılarak üretilmektedir. Bu cihazlar, çeşitli uygulamalar için gerekli olan aritmetik işlemin gerçekleştirilmesi için geniş bir alan sağlamaktadırlar. Hesaplamalar işaretli veya işaretsiz tamsayılarda

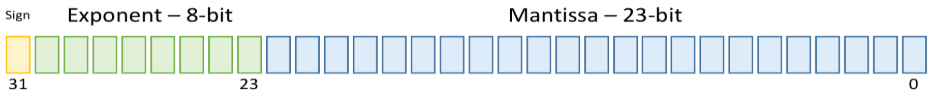
yapılabilir ve 32 bit bu işlemler için iyi bir değer aralığı verir: sırasıyla (+/- 2) milyar veya işaretsiz sayılarda ise 4 milyar'a kadar çıkmaktadır.

Şimdi ise floating point kavramının işlemcide ele alınma konseptine gelirsek, değer iki tamsayı olarak saklanır: bunlar mantis (ondalık kısım) ve üs. Temsil edilen sayı, üssün gücüyle 2 ile çarpılan mantistir. Tipik olarak, bu iki tam sayı 32 bitlik bir sözcükte bit alanlarında saklanır, ancak daha yüksek hassasiyetli varyantları da mevcuttur. En yaygın biçim IEEE 754-1985 standartıdır. Floating point kullanmanın sağladığı en iyi fayda, temsil edilebilecek geniş değer aralığıdır, ancak bu, kodlama ve bazı ödünleşmeler sırasında ekstra bakım maliyetiyle gelir.[39]

Performans: Kayan nokta işlemleri tamsayılara göre çok zaman alır. İşlem yazılımda yapılırsa, yürütme süresi gerçekten çok uzun olabilir. Donanım kayan nokta birimleri işlemleri makul ölçüde hızlandırır.

Hassaslık (Precision): Değerlerin kayan noktada gösterilme biçimi nedeniyle, bir değer tam olarak beklediğiniz gibi olmayabilir. Örneğin, 4.0 değerine sahip bir değişken bekleyebilirsiniz, ancak gerçekte 3.999999 olabilir.

Kayan nokta değeri şekil 4.4'de görüldüğü gibi üç alanla temsil edilir: işaret (1 bit), üs (8 bit) ve mantis (23 bit):



Şekil 4.4 : Floating Point Bit Gösterimi [39].

Bir sayıyı yukarıda ki formda göstermek için aşağıdaki adımlar takip edilir:

- Mantis alanı bir değere ayarlanmıştır, böylece ikili noktadan önce her zaman 1 olur, bu nedenle atlanır, böylece ekstra bir hassasiyet elde edilir. (0 değeri bir istisnadır.)
- İşaret biti, pozitif sayılar için 0 veya negatif için 1 olarak ayarlanır.
- Üs, gerekli üs artı 127 bias olan bir değere ayarlanır.

Bu adımları bir örnek ile gösterelim;

Şimdi 10 tabanında 14.25 sayısını ele alalım. 2'lik taban karşılığı 1.11001'dir. 2'lik tabanda yeniden yazarsak, 1.11001 * 2^3 formatını elde ederiz.

- Mantis değerimiz .11001

- Üs değeri bias olan 127 üzerine eklenir. Onluk tabanda 130 elde edilir. (ikilik tabanda 10000010)
- İşaret biti pozitif olduğu için 0 verilir.

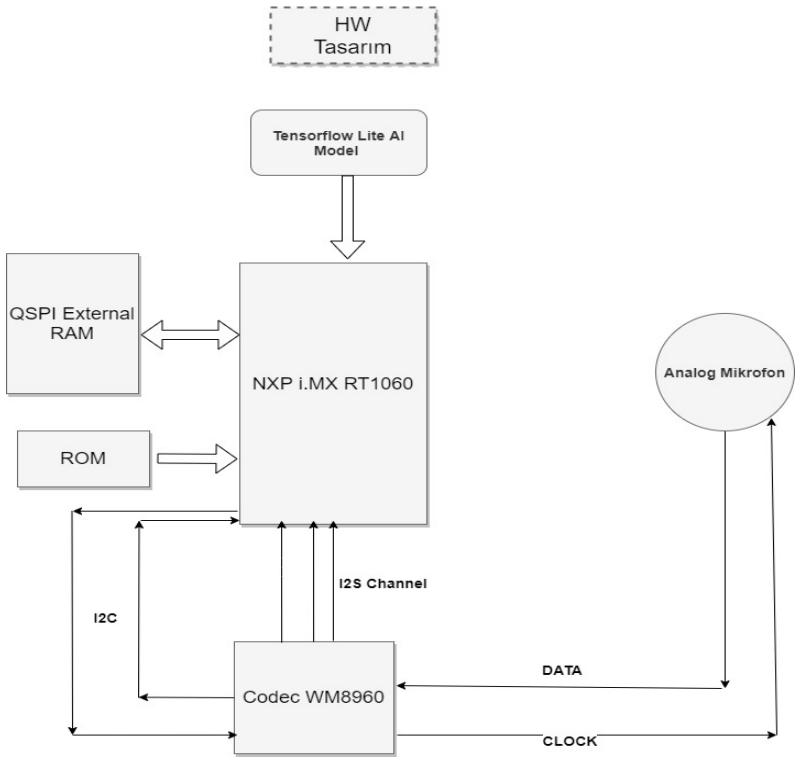
Elde edilen kayan nokta (floating point) gösterimi denklem 4.1'de ki gibi elde edilir:

0 | 1000 0010 | 110 0100 0000 0000 0000 0000 (0x41640000) (4.1)

Yukarıdaki bilgiler ışığında kayan nokta gösterimi sayısal hesaplamayı çok daha kolay hale getirir. Gömülü programlamada, her sayıda ondalık (veya ikili) noktanın "gerçekten" nerede olduğunu bilinmelidir. Bazen bir programın birkaç farklı sayı aralığını ele alması bile gerekebilir. Floating point yapısı yazılımı tamsayılar veya sabit nokta gösterimleri kullanarak yazmaktan kurtarır. Bundan dolayı floating point işlemler "Gömülü Sistemler Mimarisi Tasarımı" başlığı altındaki en önemli nokta olacak ve algortimanın gömülü sistem yazılımına dönüştürüldüğü aşamalarda da üzerinde de durulacaktır.

4.2 Hardware Mimarisi

Tez kapsamında kullanılan MIMXRT 1060 Eval Board mimarisi yapısı blok diyagram haliyle Şekil 4.5'de gösterilmiştir.



Şekil 4.5 : MIMXRT 1060 EVAL Board HW Mimarisi.

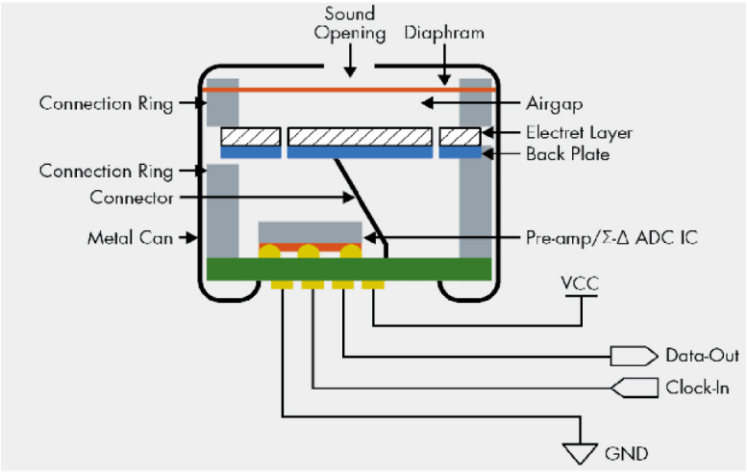
Bu mimari ışığında mikrofon yapıları üzerinde durulacak, Mikrofondan alınan datanını codec'te işlenmesi ve I2S kanalıyla işlemciye iletilmesi aşaması ele alınacak. Devam eden doğrultuda SAI blokları vasıtasıyla datanın işlenmesi, flash ve ram optimizasyonu üzerinde durulacaktır.

4.2.1 Mikrofon

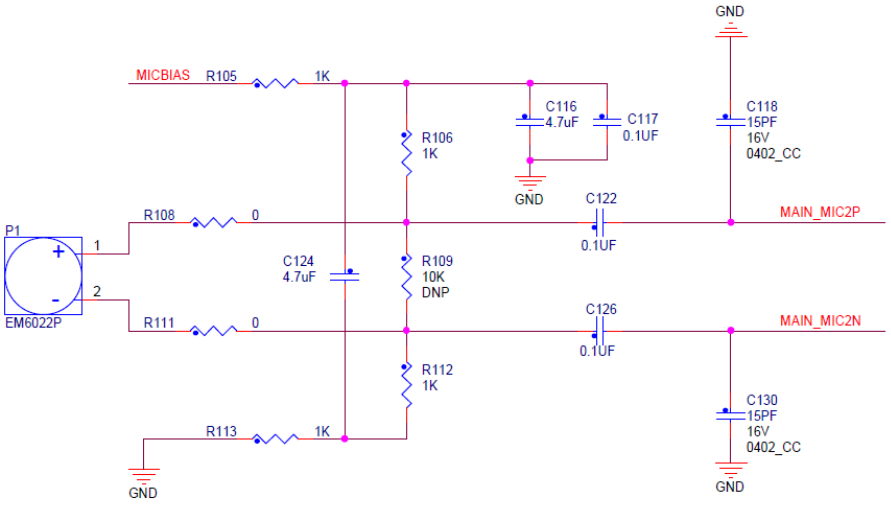
En basit tanım ile, bir mikrofona dönüştürücü (tranducer) yapısı denir ve ses tarafından üretilen ses dalgalarını elektrik enerjisine dönüştürebilir. Tüm mikrofon yapıları, ses dalgalarına yanıt vermek için çeşitli şekillerde yapılmış diyaframlara veya benzer öğelere sahiptir. Mikrofona giren ses dalgası diyaframa çarpar ve ses basıncındaki değişime göre diyafram içe veya dışa hareket eder ve mekanik titreşim oluşturur. [40]

Aynı şekilde basitçe trasducer yapısının tanımını yapacak olursak; bir enerji biçimini bir başka enerji biçimine dönüştüren cihazlara verilen isimdir. Tranducer'lar kullanıldıkaları malzemeye göre isimlendirilirler. Tez kapsamında geliştirme kartı üzerinde bir adet Analog ECM EM6022P mikrofunu kullanılmıştır.

Elektret kondansatörlü mikrofon (ECM mikrofon), çalışma için kapasitör şarj / deşarj ilkesini uygular. Diyafram bir kapasitörün bir plakası gibi davranır, ses nedeniyle titreşim üretilir, daha sonra sinyal iletimini yapmak için plakalar arasındaki yük değişikliklerini üretir. Titreşim tamamen elektromekanik bir mekanizma olduğundan, genellikle harici taşınabilir mikrofonda kullanılan tüm akustik frekans tepkisine (0-20 kHZ) ve sinyal iletiminde düşük bozulmaya sahiptir. Şekil 4.6'da ECM bir mikrofonun iç yapısı görsel olarak verilmiştir. Şekil 4.7'de mikrofonun devre şematiği verilmiştir.

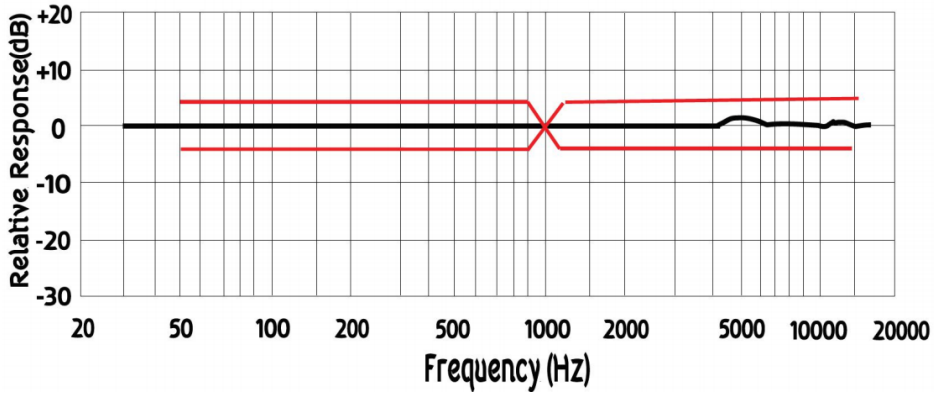


Şekil 4.6 : ECM mikrofon iç yapısı.



Şekil 4.7 : Mikrofon devresi.

Şekil 4.8'de frekans - yanıt grafiği verilmiştir.



Şekil 4.8 : EM6022P mikrofonuna ait frekans -response grafiği [41].

4.2.2 Codec yapısı ve HW mimarisinde kullanımı

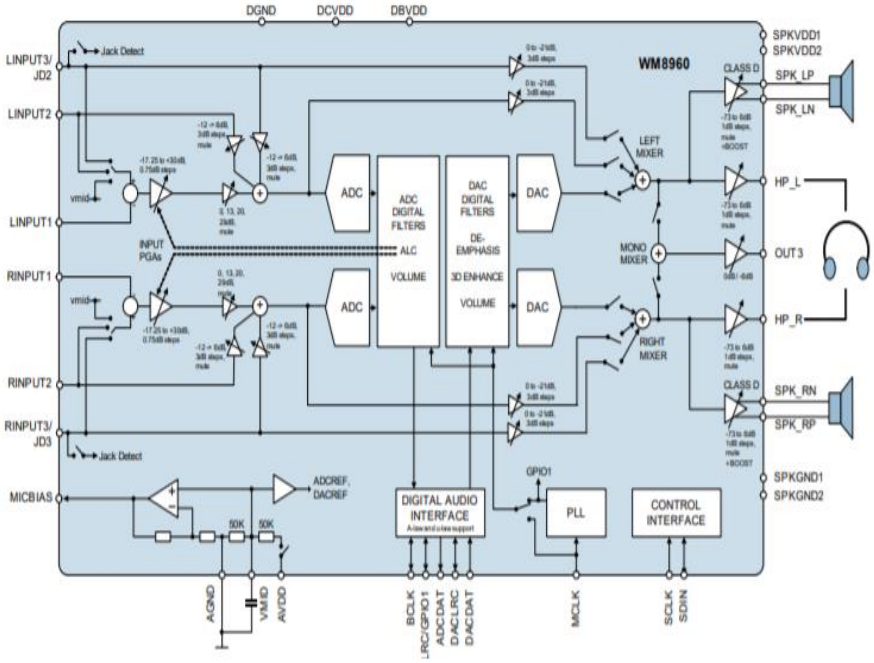
En basit tanımıyla codec, donanımda, analog sesi dijital sinyaller olarak kodlayan ve dijitalin tekrar analog olarak kodunu çözen tek bir aygıtı ifade etmektedir. Başka bir deyişle, hem analog-dijital dönüştürücü (ADC) hem de aynı dijital-analog dönüştürücü (DAC) içerir. Tez kapsamında kullanılan CODEC, bir mikrofonun aldığı sesi değiştirmek ve iletmek için 0s ve 1s akışına (analogdan dijitale dönüştürme) veya dijital sinyal iletilirse daha yaygın bir "veri akışı" na dönüştürmekten sorumludur.

Tez kapsamında kullanılan geliştirme kartı üzerinde Cirrus Logic firmasının WM8960 ürünü kullanılmıştır.

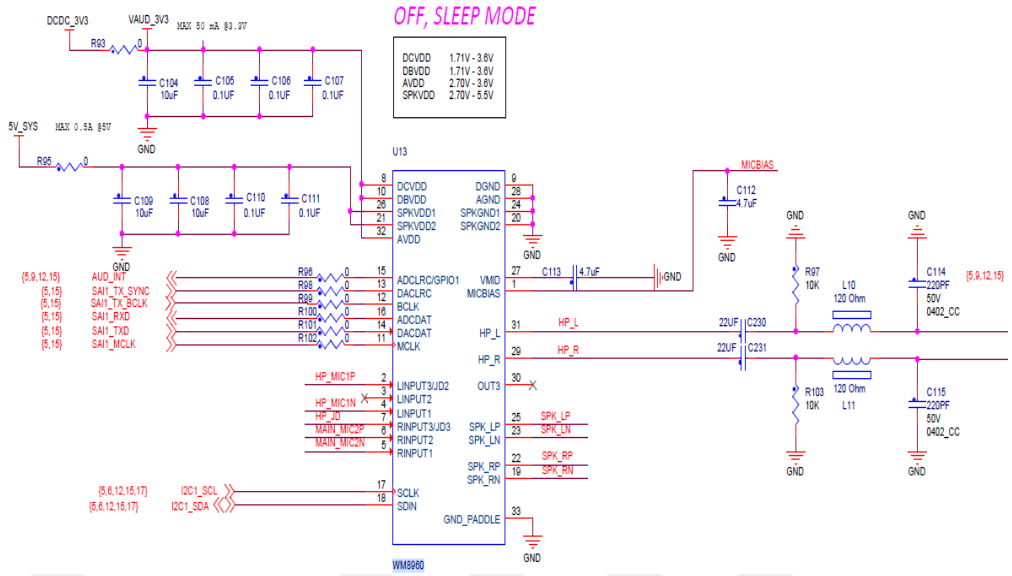
Bu entegre, verimliliği ve güç tüketimini optimize etmek için çift modlu şarj pompası mimarisi içeren stereo toprak referanslı kulaklık amplifikatörlerine sahip yüksek performanslı, ultra düşük güçlü bir stereo codec bileşenidir. Toprak referanslı kulaklık ve hat çıkışları AC bağlantı kapasitörlerini ortadan kaldırır ve her iki çıkış da toprak gürültüsünü reddetmek için ortak mod geri besleme yolları içerir. Şekil 4.9'da Codec iç devre yapısı gösterilmiştir. Çizelge 4.2'de datasheet değerleri görülmektedir.

Çizelge 4.2 : WM8960 Parametrik Özellikler.

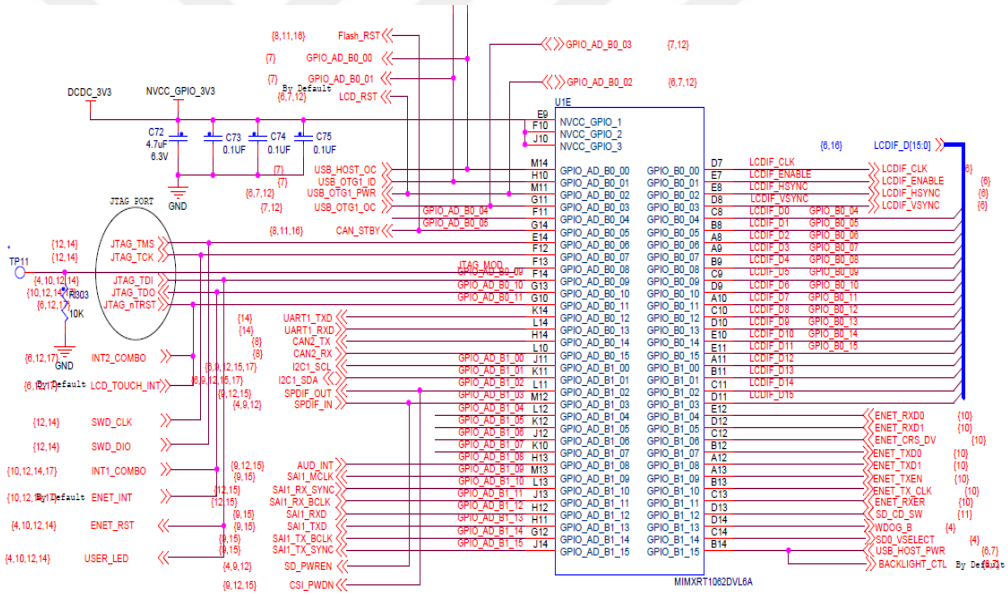
Datasheet Bilgileri	Veri Değerleri
Channels In	2
Channels Out	2
DAC THD+N (dB)	-84
ADC THD+N (dB)	-82
Analog Power Supply (V)	2.7 – 3.6
Digital Power Supply (V)	1.71 – 3.6
Logic Power Supply (V)	2.7 – 3.5
Power Consumption (mW)	5-2,868.2



Şekil 4.9 : Codec devresi iç blokları [42].



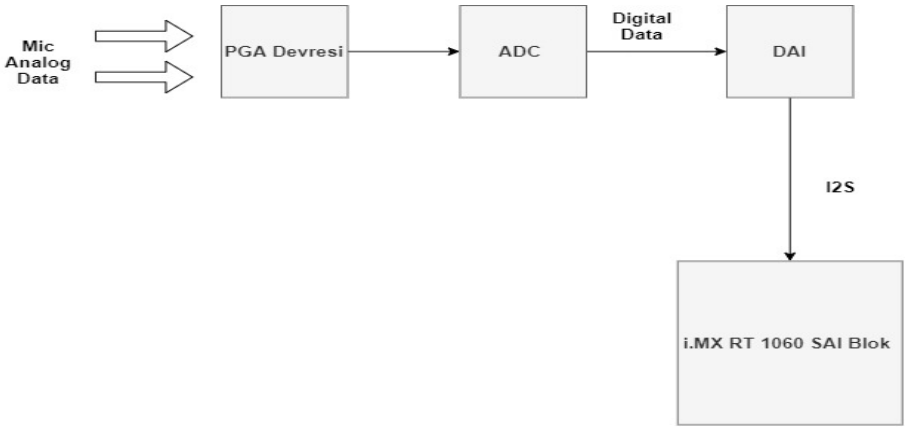
Şekil 4.10 : Codec Yapısı.



Şekil 4.11 : Ana İşlemci Yapısı.

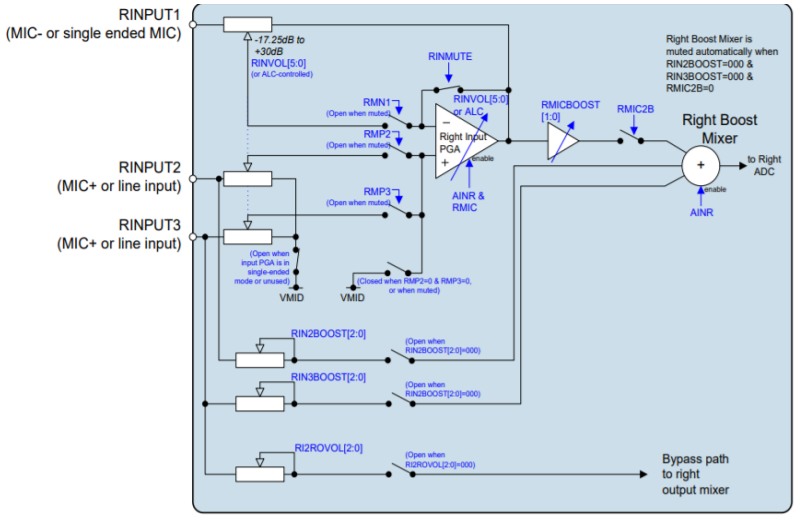
Sistemin çalışma mimarisi şu ededir;

Şekil 4.10'de analog mikrofonundan MAIN_MIC2P ve MAIN_MIC2N çıkışları Şekil 4.11'de görüleceği üzere Codec yapısının RINPUT1 ve RINPUT2 girişlerine gitmektedir. Mikrofon outputlarından gelen datalar sırasıyla, PGA, ADC ve DAI bloklarında işleme alınır ve I2S protokolü ile i.MX RT1060 işlemcisine iletilirler. Bu akış şekil 4.12'de görülebilmektedir.



Şekil 4.12 : Mikrofon Analog Data işleme blokları.

WM8960'da mikrofonlar LINPUT1 ve LINPUT2 veya LINPUT3 arasında ve RINPUT1 ve RINPUT2 veya RINPUT3 arasında bağlanabilir. Alternatif olarak, tek uçlu mikrofonlar LINPUT1 veya RINPUT1'e bağlanabilmektedir (Şekil 4.11). Tek uçlu mikrofon giriş yapılandırmasında mikrofon sinyali LINPUT1 veya RINPUT1'e girilmeli ve giriş PGA'nın dahili evirmeyen girişi VMID'ye geçirilmesi gerekmektedir. Diferansiyel modda, daha büyük sinyal sol kanalda LINPUT2 veya LINPUT3'e veya sağ kanalda RINPUT2 veya RINPUT3'e girilmelidir. Daha küçük (örneğin gürültülü toprak bağlantısı) LINPUT1 veya RINPUT1'e girilmelidir. Mikrofon PGA'larının kazancı doğrudan yazılım aracılığıyla veya ALC / Sınırlayıcı kullanılarak kontrol edilebilir. Diferansiyel mikrofon konfigürasyonunda evirici olmayan giriş olarak çalışırken LINPUT2, RINPUT2, LINPUT3 ve RINPUT3 girişleri destek mikserine veya baypas yoluna bağlanmamalıdır. Şekil 4.13'de PGA kazanç devresi verilmiştir.



Şekil 4.13 : PGA Kazanç Devresi [42].

PGA (Programmable Gain Amplifier), kullanıcı tarafından programlanabilen kazanca sahip opamp tabanlı, evirmeyen bir amplifikatör uygular. Bu amplifikatör yüksek giriş empedansı, geniş bant genişliği ve seçilebilir giriş voltaj referansına sahiptir. Bu devrenin çıktısı olarak elde edilen sinyal ADC kanalının girişi olarak kullanılır. PGA devresi aşağıdaki register bitleri vasıtası ile konfigure edilmetedir.

AINL, AINR register bits (Boost Mixer Enable)

LMIC, RMIC register bits (Input PGA Channel Definition)

LINVOL, RINVOL register (INPUT PGA volume controls)

LINMUTE, RINMUTE register (INPUT Boost)

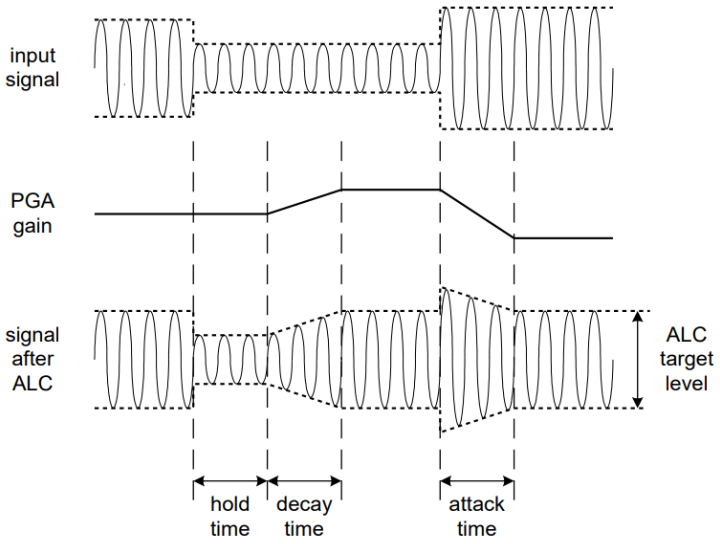
MICBIAS register (Microphone Bias)

4.2.3 ADC yapısı ve HW mimarisindeki yeri

WM8960 entegresinde, stereo 24 bit, 64x aşırı örneklenmiş sigma-delta ADC'ler kullanır. Çok bitli geri besleme ve yüksek örnekleme hızlarının kullanılması titreşim ve yüksek frekanslı gürültünün etkilerini azaltır. ADC Tam Ölçekli giriş seviyesi AVDD ile orantılıdır. 3.3V besleme gerilimi ile tam ölçek seviyesi 1.0Vrms'dir. Tam skalanın üzerindeki herhangi bir voltaj ADC'yi aşırı yükleyebilir ve bozulmaya neden olabilir. ADC'lerin çıkışı 0,5dB'lik adımlarla –97dB ila + 30dB aralığında dijital olarak yükseltilebilir veya zayıflatılabilir. Her kanalın ses seviyesi ayrı ayrı kontrol edilebilir. ADC filtreleri, ham çok bitli aşırı örneklenmiş verileri ADC'den dijital ses arabiriminde çıkacak doğru örnekleme frekansına dönüştürmek için gerçek 24 bit sinyal işleme gerçekleştirir.

Automatic level control (ALC)

WM8960, giriş sinyali seviyesinden bağımsız olarak sabit bir kayıt ses seviyesini korumayı amaçlayan otomatik bir seviye kontrolüne sahiptir. Bu, ADC girişindeki sinyal seviyesi sabit kalacak şekilde PGA kazancını sürekli olarak ayarlayarak elde edilir. ADC çıkışını izlenir ve gerekirse PGA kazancını değiştirir. ALC işlevi etkinleştirildiğinde, 0 ve 1 kayıtlarının (LINVOL, IPVU, LIZC, LINMUTE, RINVOL, RIZC ve RINMUTE) ayarları yok sayılmaktadır. Şekil 4.14’de ALC operasyonları gösterilmiştir.



Şekil 4.14 : ALC Operasyonları.

Bu noktada hold time, decay time ve attack time terimlerinin tanımın yapmak gerekirse;

Hold time, yani bekletme süresi, hedeflenen hedefin altında olan tepe seviyesi ile yükselmeye başlayan PGA kazancı arasındaki gecikmedir. 2'nin üssü bir güçle programlanabilir, Örnek vermek gerekirse; 2.67ms, 5.33ms, 10.67ms vb. 43.7s'ye kadar. Alternatif olarak, tutma süresi de sıfıra ayarlanabilir. Bekletme süresi sadece kazanç artışı için geçerlidir, sinyal seviyesi hedefin üzerinde olduğunda kazanımı azaltmadan önce gecikme olmaz.

Decay time, yani kazanç artışı süresi, PGA kazancının aralığının% 90'ında yükselmesi için geçen süredir (örneğin –15B'den 27.75dB'ye). Dolayısıyla, kayıt seviyesinin hedef değerine dönmesi için geçen süre, hem bozulma süresine hem de gereken kazanç ayarına bağlıdır. Kazanç ayarı küçükse, bozulma süresinden daha kısa olacaktır. Bozulma süresi, 24ms, 48ms, 96ms, vb. İla 24.58 saniyeye kadar ikinin üssü ölçüsünde güçle programlanabilir.

Attack time, yani kazanç azaltma süresi, PGA kazancının aralığının % 90'ında rampa yapması için geçen süredir (örneğin, 27,75 dB'den -15B kazanımına). Bu nedenle, kayıt seviyesinin hedef değerine dönmesi için geçen süre hem attack süresine hem de gereken kazanca ayarlamasına bağlıdır. Kazanç ayarlaması küçükse, saldırı süresinden daha kısa olacaktır. Attack süresi, 6ms, 12ms, 24ms, vs.'den 6.14s'ye kadar ikinin üssü ölçüsünde güçle programlanabilir.

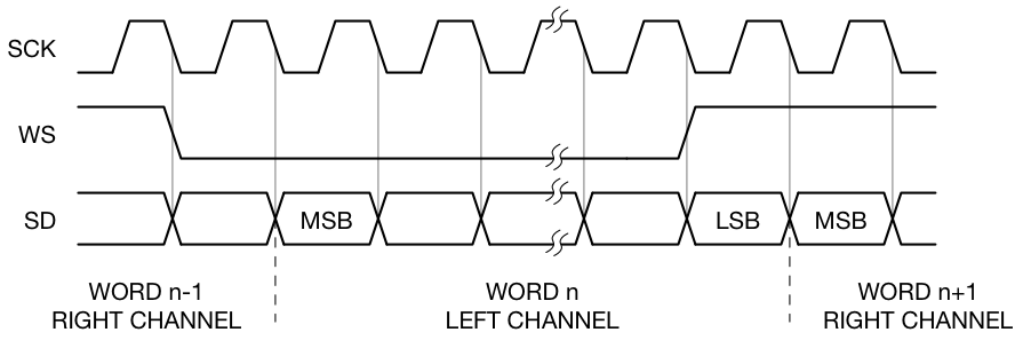
4.2.4 DAI (Digital Audio Interface) HW mimarisindeki yeri

DAI, bu blok üzerinden sayısal ses datası I2S veri yolu kullanılarak DSP işlemciye gönderilmektedir. WM8960'a DAC verisi girmek ve ondan ADC verisi çıkarmak için kullanılır.

Saat sinyalleri, backprop, WM8960 bir master olarak çalıştığında veya bir slave olduğunda girilebilir. ADCLRC, bir GPIO pini olarak da yapılandırılabilir. Bu durumda, ADC bir çerçeve saati olarak DACLRC'yi kullanır. ADCL etkinken ADCLRC / GPIO1 pin işlevi değiştirilmemelidir.

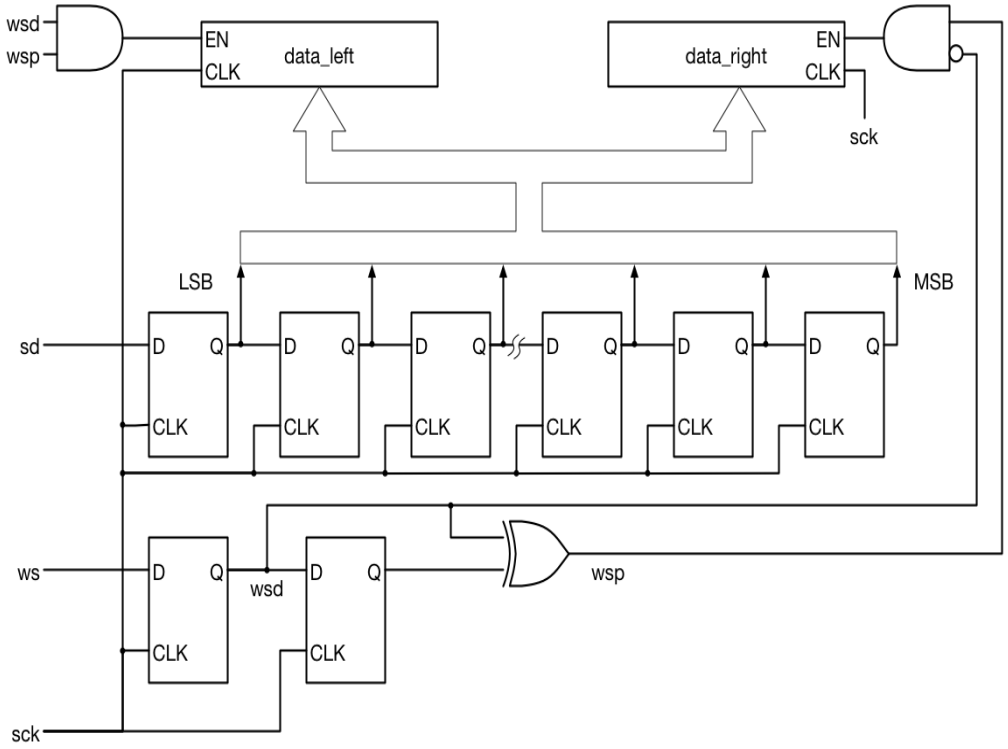
I2S Veri yolu

I2S (Inter-IC (Integrated-Chip) Sound olarak da bilinir), tek bir PCB üzerindeki dijital ses cihazları arasında seri veri aktarımı için özel olarak geliştirilmiştir. Bazı MCU'lar, uygun ses CODEC'leri ve ADC / DAC'lerle arabirim oluşturmak için bu protokolü kullanmalarını sağlayan özel I2S donanım bağlantı noktalarına sahiptir. I2S bir stereo ses verisi akışı gönderir. Her ses örneği için bir sol kanal ve bir sağ kanal değeri vardır. En yaygın değerler 16, 20, 24 ve 32 bit değerler olmakla birlikte, değerler herhangi bir sayıda bit olabilir. Ses verilerinin gönderilmesinde üç sinyal vardır. Bir seri saat SCK, verilerin sol kanal veya sağ kanal için olup olmadığını gösteren bir kelime flaş sinyali WS (Word Select) ve bir seri veri sinyali SD (Serial Data) . WS ve SD sinyalleri SCK sinyali ile senkronize olur ve SCK'nın düşen kenarında değişir. Önce veri en anlamlı bit olarak gönderilir. Bir kırışıklık, verinin kelime strobuna göre bir döngü geç olmasıdır. Zamanlama şeması ayrıntıları 4.15'da gösterilmiştir.



Şekil 4.15 : I2S Protokol Zamanlama Diyagramı.

Şekil 4.16'da receiver diyagramı yer almaktadır.

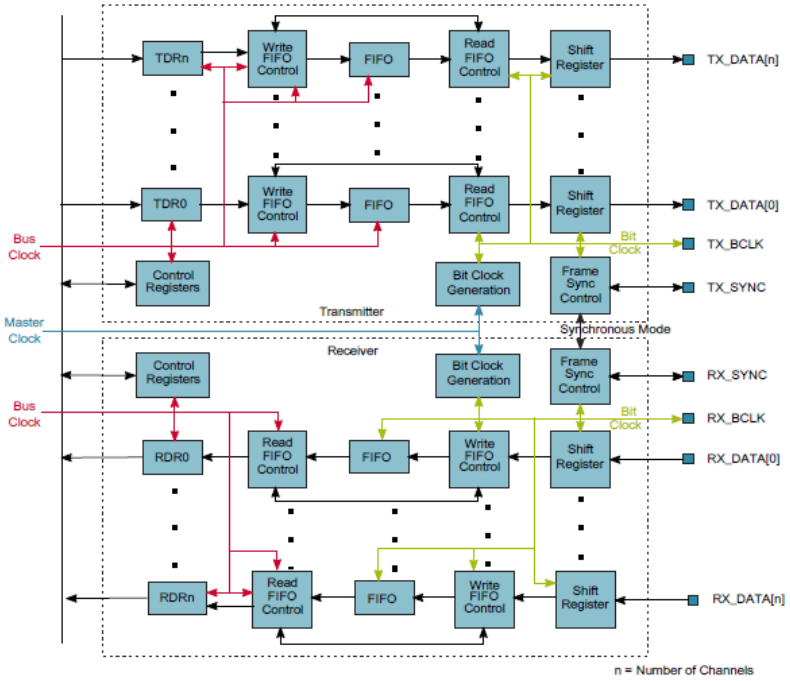


Şekil 4.16 : I2S Receiver Diyagram.

I2S (veya I2S) modülü, I2S, AC97, TDM ve codec / DSP arayüzleri gibi frame senkronizasyonlu fullduplex seri arayüzleri destekleyen senkron bir ses arayüzü (SAI) sağlar. SAI modülünün içerdiği özellikler aşağıdaki gibi sıralanmıştır:

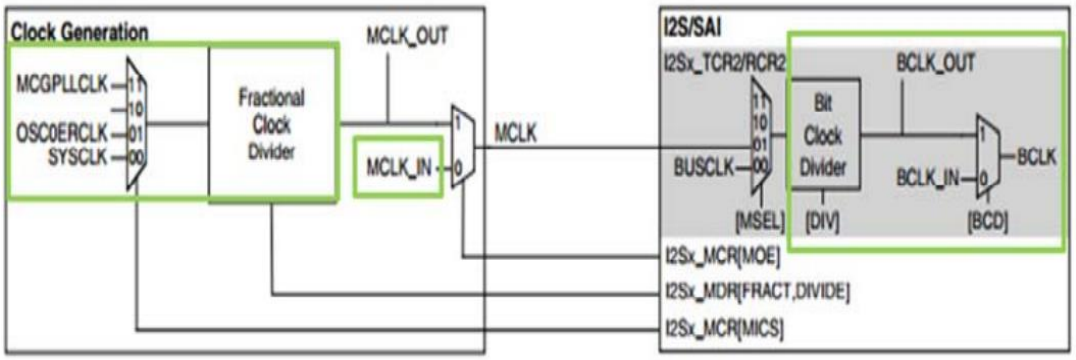
- Örnek boyutu olarak da bilinen kelime boyutu değişkendir ve 8 ila 32 arasında değişebilir. kelime boyutu olarak da bilinen kelime boyutu değişkendir ve 8 ila 32 bit arasında değişebilir.
- İlk kelimenin kelime boyutu, bir karede kalan sözcüklerden ayrı olarak yapılandırılabilir.
- SAI modülü, kesme veya modül DMA (Direct Memory Access) isteklerinin sayısını azaltan 8 kelimelik ilk giriş ve ilk çıkış arabelleğini veya FIFO'yu destekler.
- Senkron verici ve alıcı, aynı örnekleme frekansında oynatma ve kayıt içeren uygulamalar için pin çıkışını azaltır.
- SAI modülü 32 kelimeye kadar işleyebilir ve çok kanallı ses uygulamasına izin verir.

Şekil 4.17'de ise I2S/SAI blok diyagram çalışma şekli gösterilmiştir.



Şekil 4.17 : I2S/SAI Blok Diyagram.

SAI modülü hem senkronize hem de asenkron çalışmaya izin veren verici ve alıcı için bağımsız saatlere sahiptir. SAI modülünü kullanarak veri aktarırken, FIFO gönderilmesi gereken sözcükleri tutar. Sözcükler, her yeni çerçeve başlatıldığında gönderilmektedir. Yeni bir çerçeve başladığında kelimeler gönderilir. Yeni bir karenin başlangıcında, kaydırma yazmacı FIFO'dan kelimeler alır ve bit saatine göre göndermeye başlar. SAI modülünü kullanarak veri alırken, yeni bir çerçevenin başlangıcında, alıcı, bit saatine göre alınan verileri örnekleyecektir. Örneklenen veriler daha sonra FIFO'da saklanacaktır. Şekil 4.18'de saat frekans çalışma yapısı verilmiştir.



Şekil 4.18 : SAI Saat Frekans Çalışma Yapısı.

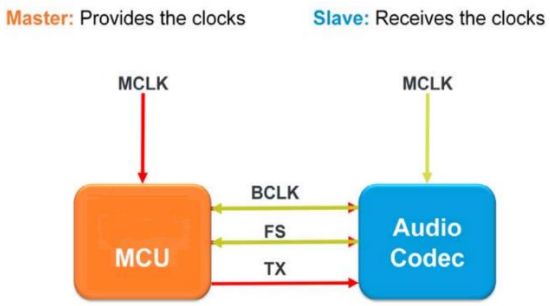
SAI'nın ana saati dahili veya harici olabilmektedir. Harici bir saat kaynağı için, (MCLK_IN) sinyalindeki bir ana saat kullanılır. Harici bir saat kaynağı kullanmanız durumunda, ses örnekleme frekansının katlarından birden fazla frekansa sahip özel bir osilatör beklenir. Ses örnekleme frekansından katları beklenir.

I2S protokolünü uygulamak için aşağıdaki sinyaller gereklidir:

Frame sync: Bu sinyal, yeni bir veri çerçevesinin ne zaman başlayacağını belirtmek için gereklidir. Frame senkronizasyon sinyali, ses örnekleme frekansında bir saattir. Veriler kare senkronizasyonunun düşen kenarında gönderilir Bu sinyalin kare senkronizasyonunun düşen kenarında da duyabilirsiniz. Bu sinyali diğer satıcılar tarafından çerçeve saati, kelime seçimi veya kelime saati olarak da anlayabilirsiniz.

Bit Clock (Serial Clock) : Bu saat, seri verileri göndermek veya almak için kullanılır. Hız olacak örnek büyüklüğüne bağlıdır. Örneğin 32 bit olduğu varsayıldığında, seri saat m 64 kare s nc'dir Hem kare nc hem de bit saati kare senkronizasyonunun 64 katı olmalıdır. Hem çerçeve senkronizasyonu hem de bit saati sürekli sinyallerdir. Şekil 4.19 da haberleşme modeli verilmiştir.

Data hatları: RX ve TX hatları. Son olarak, tüm saatler ana saat adı verilen bir referans kullanmalıdır Bu saat frame senkronizasyonunun 256, 384 veya 512 katı olabilir.



Şekil 4.19 : I2S Haberleşme Modeli.

I2S iletişim protokolü iki cihaza ihtiyaç duyar: Master ve Slave. Master cihaz, slave'e çerçeve senkronizasyonu ve seri saat sağlar.

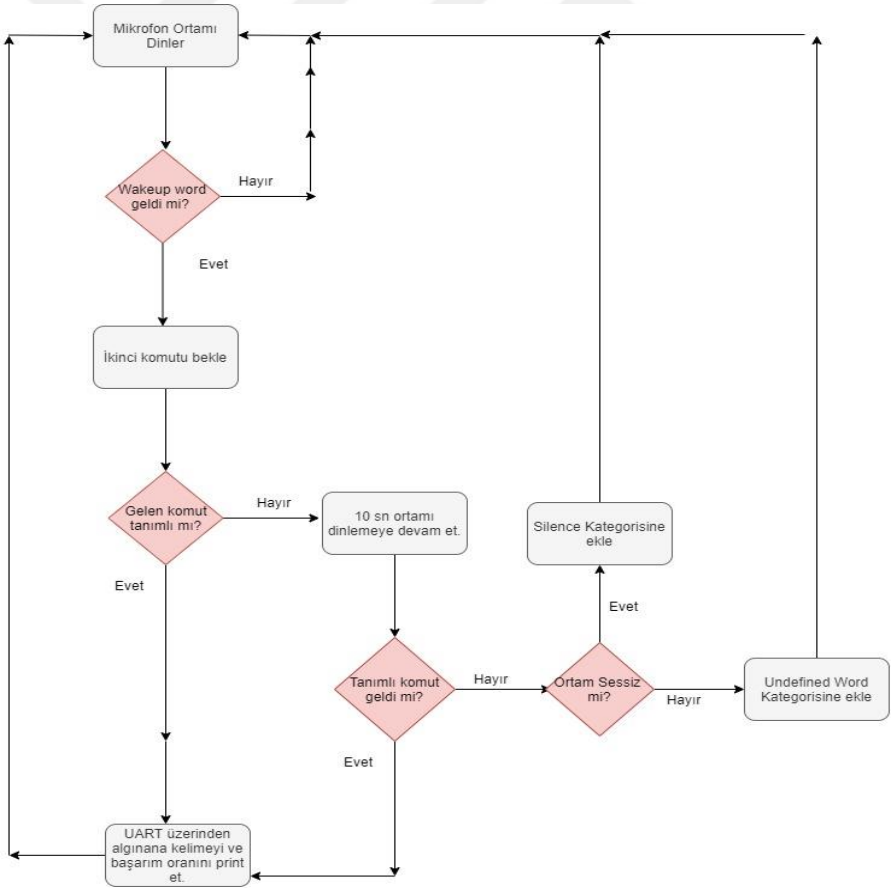
Tez kapsamında yapılan çalışmada, i.MX RT1060 MCU master ve ses codec bileşeni olan WM8960 ise slave'dir.

4.2.5 HW mimarisi çıktıları

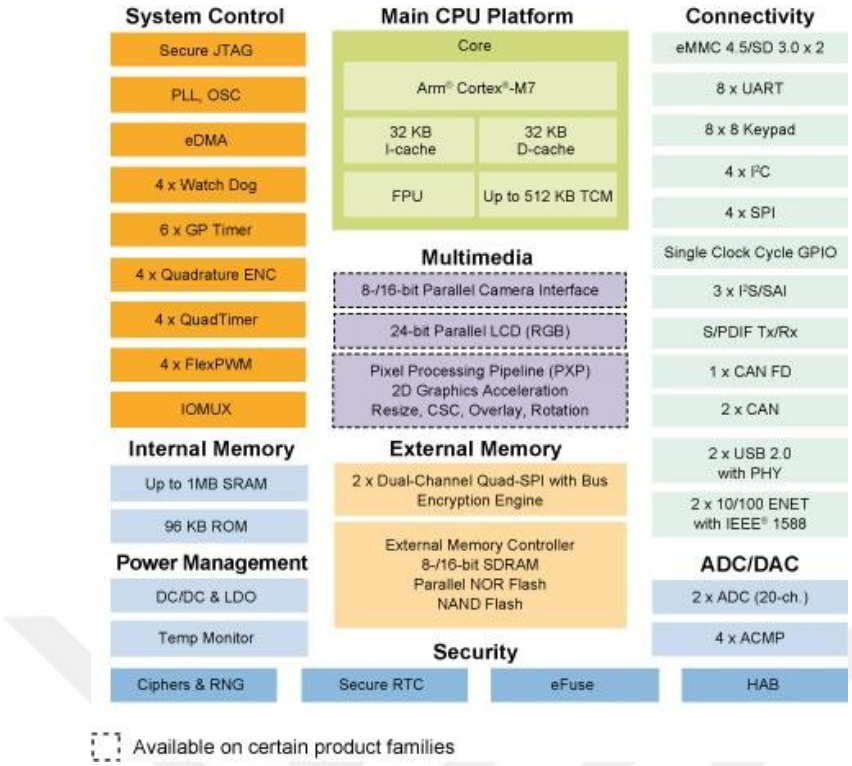
Bu alt başlık kapsamında ortamdaki ses datasının mikrofonlar vasıtası ile algılanıp, elde edilen ses sinyalinin işlemci donanım mimarisinde hangi işlemlerden geçtiği anlatılmaya çalışılmıştır. Bu sebeple mikrofon yapıları, codec yapısı ve I2S veri yolu kavramları detaylı bir şekilde anlatılmış ve PCB şematik dosyaları ve görseller yardımıyla zenginleştirilmiştir.

4.3 SW Mimarisi

Bu başlık altında donanımsal mimarinin programlanması, elde edilen datanın anlamlandırılması, proje yazılım ağacının oluşturulması, Tasarlanan TFLite modelinin gömülü yazılıma entegre edilmesi, başarım çıktılarının UART vasıtasıyla ekrana print edilmesi gibi yapılan işlemler detaylandırıması üzerinde durulacaktır. SW mimari akış diyagramı şekil 4.20'de ve blok diyagramı şekil 4.21'de gösterildiği gibidir.



Şekil 4.20 : Yazılım Akış Şeması.



Şekil 4.21 : i.MX RT 1060 Blok Diyagram [43].

Yazılım mimarisini açmamız gerekirse; yapı 5 ana kola bölünmüştür. Bu kollara değinecek olursak Driver, MFCC Computing, TFLite Deep Learning Model, Wakeup Word Detection, KWS katmanlarıdır.

4.3.1 Driver katmanı

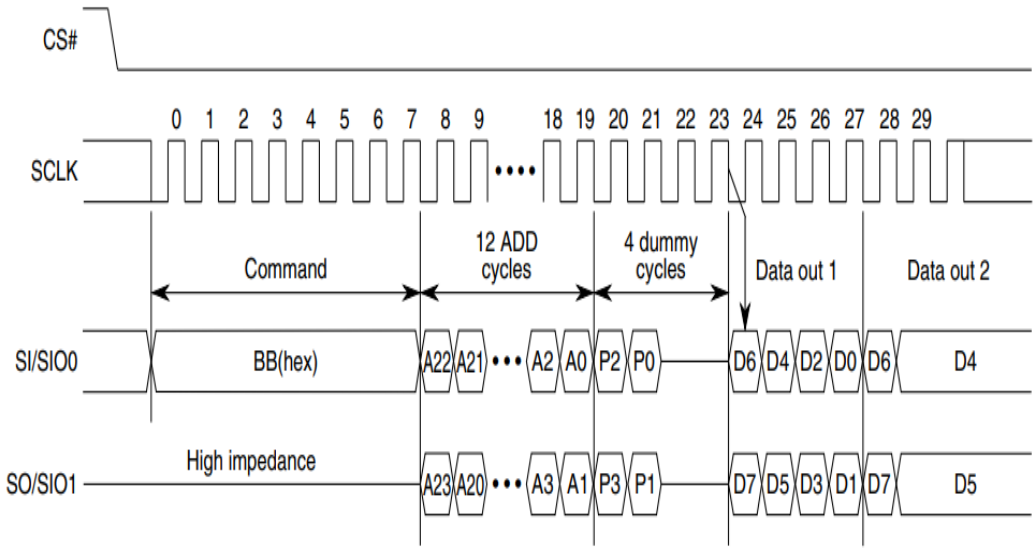
Bu katman işlemcinin driver ayarlanının yapıldığı katmandır. Clock, gpio, i2c, spi, uart ve sai modüllerinin register ayarları bu katmanda yapılır. İlgili modüller istenilen özelliğe işlemci datasheet'in de yazan register adresleri ve komut listerine göre programlanır. Burada ilgili modüllerin görevlerini kısaca yazmak gerekirse;

fsl_gpio.c ve fsl_gpio.h : Genel input/output register'larinin ayarların yapıldığı driver işlemlerini içeirir.

fsl_clock.c ve fsl_clock.h : Sistem clock frekans ayarlarının yapıldığı driver işlemlerini içerir.

fsl_Ipi2c.c ve fsl_Ipi2c.h : Slave Codec modülünün register bitlerinin kontrol etmek için yazılmış veri yolu driver işlemlerini içerir.

fsl_spi.c ve fsl_spi.h: External ram'e veri yazma, veri okuma amacıyla QSPI driver ayarlarının yapıldığı driver işlemlerini içerir. Şekil 4.22'de QSPI protokolü görülebilmektedir.



Şekil 4.22 : QSPI data hattı [44].

fsl_luart.c ve fsl_luart.h: Teraterm üzerinden ilgili dataları ekrana basmamızı sağlayacak registerların set edildiği driver işlemlerini içerir.

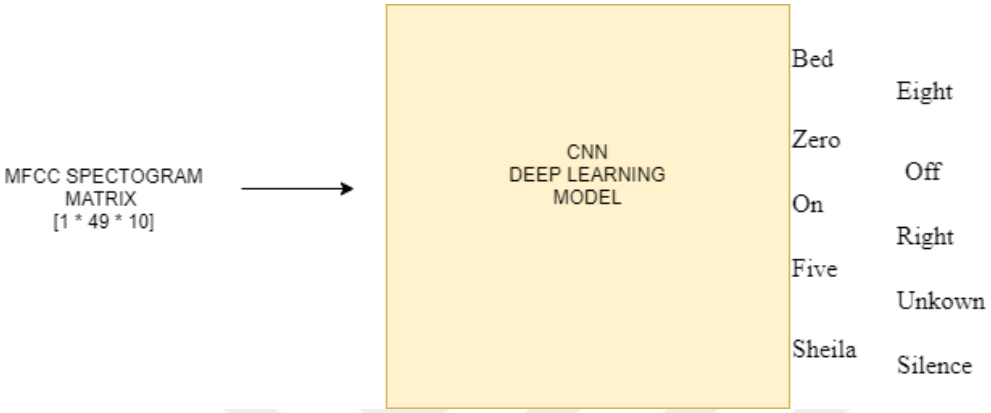
4.3.2 MFCC hesaplamaları

Bu katman i2s veri yolu ile işlemciye alınmış ortam sesinin MFCC spektogram matrisinin elde edilmesi aşamasını içerir. Bu katmanda floating point kurallarına uyularak Deep Learning Model katmanına girişi olacak speoctogram matrisi elde edilir. Bu aşamada yapılan işlemler, windowing (pencereleme) fonksiyonu, normalizing, FFT, power spectrum, Mel filtre bankası uygulaması, DCT matris uygulaması adımlarıdır. Çalışmanın MCUXpresso idesinde yazılmış kod parçalarına pencereleme fonksiyonu için EK 4-A, Normalizasyon işlemi için EK 4-B, FFT ve Power Spektrum işlemleri için EK 4-C, Mel Fitre bankasının oluşturulması aşaması için EK 4-D VE DCT matris uygulaması için EK 4-F'de ki görseller incelenebilir.

4.3.3 TFLite deep learning model

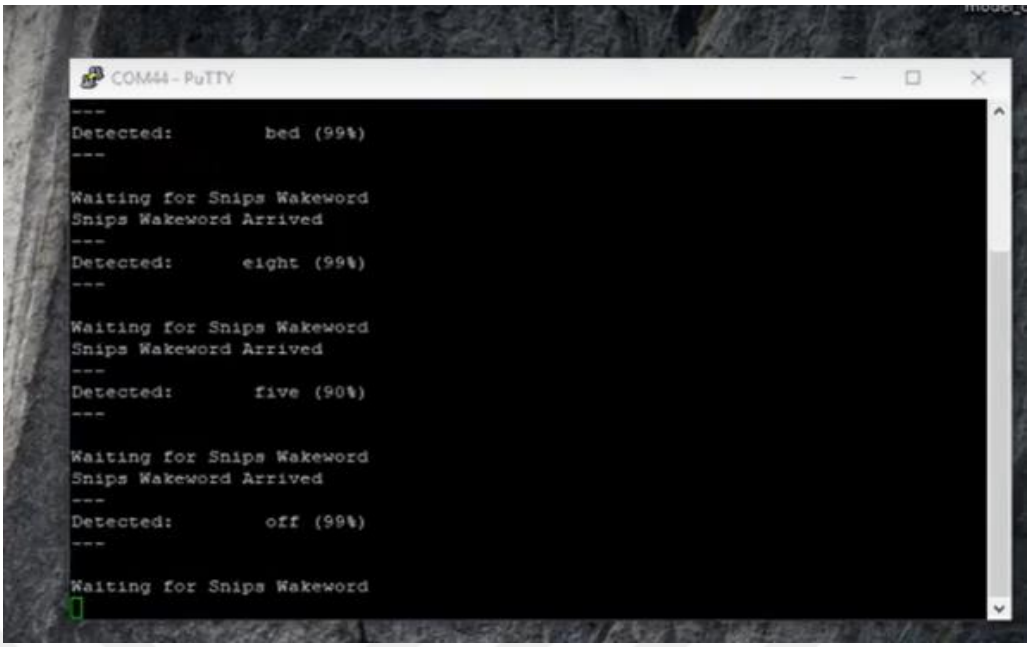
Tensorflow kütüphanesi ile hazırlanıp TFLite versiyonuna dönüştürülmüş olan header dosyasının embedded koda impelmente edilmesi aşamasını içerir. İlgili header file

proje dosyasına ekleni ve ilgili yerde çağırılır. Daha sonrasında model parametrelerinin tanımlandığı InferenceInit() fonkisyonu içerinde ilk değer verilir. RunInference() fonksiyonu ile de çağırılarak gömülü sistem mimarisi ile elde edilen ses verisini modele göre etiketler. Şimdi ise embedded sistem içerisinde bu sürecin nasıl gerçekleştiğini detaylıca görelim. Şekil 4.23 model çalışma yapısını içermektedir.



Şekil 4.23: Model Çalışması.

Yeterli bir büyüklükte bir dataset ile eğitilmiş modeli TFLite modeliyle gömülü mimariye aktardıktan sonraki çalışma adımında wakeup Word ile sleep moddan çıkan sistem gelen mfcc öznitelik spectogram matrisini eğitilmiş CNN model doğrusunda etiketler. Çıkan sonuç bir confidence struct değişkene kayıt edilir. Bu değişken bir treshold değeri ile karşılaştırılır. Belirlenen treshold değerine göre çıkan olasılık değeri o değerden küçük ise, yeniden model sorgusu yapılır. Şekil 4.24 de ise çıktılar gösterilmiştir.



Şekil 4.24: Program Çıktıları.

Ayrıca RunInference() fonksiyonun MCUXpresso ide'sinde ki kod çıktısı için EK 4-G incelenebilir.

4.3.4 Wakeup word detection

Wakeup Word (Uyandırma sözcüğü) sistemin low energy modunda kalmasını sağlar. Günlük konuşma içinde geçecek komut isimlerinin sistemi harekete geçirmesini engeller. Sistem sleep modda sadece ortamı dinler. Wakeup Word geldikten sonra ise sistem uykudan çıkar ve geçerli bir komut istemi beklemeye başlar. Wakeup Word sezinlemesi için ise ayrı bir derin öğrenme algoritması koşar. Bu algoritma ve komutların tanımlandığı model yazılım içinden paralel olarak çalışma gösterir.

4.3.5 KWS (Key Word Spotting)

Yazılımın yürütüldüğü ana katmandır. Bu katmanda tasarlanan mimari'de ilgili initization işlemleri yapıldıktan sonra, mikrofondan gelen dataları değerlenidirir ve sistemi çalıştırır. Sistem mimarisi basitçe aşağıdaki case'lerden oluşmaktadır:

```
typedef enum
{
    WakeWord_ENGINE,
    TRANSITION_TO_TF,
    TF_ENGINE,
    TRANSITION_TO_WakeWord,
    DEFAULT
}Application_States;
```

WakeWord_ENGINE, bu case altında sistem sadece ortam dinlemesi yapar ve belirlenmiş wakeup word'ün gelmesini bekler.

TRANSITION_TO_TF, bu case altında wakeup Word gelmiştir ve ardından söylenecek olan komutun algılanması için komutların etiketlendiği deep learning modele geçiş olur.

TF_ENGINE, bu case altında mikrofon vasıtasıyla gelen komutun mfcc spectogram matrisi giriş olarak alınır ve model üzerinde ilgili komuta etiketlenip, UART üzerniden print edilir.

TRANSITION_TO_WakeWord, Sistem tekrardan wakeup Word bekleme moduna geçer.

DEFAULT, Sistem bu case'e gelirse WakeWord_ENGINE'e konumuna döner.

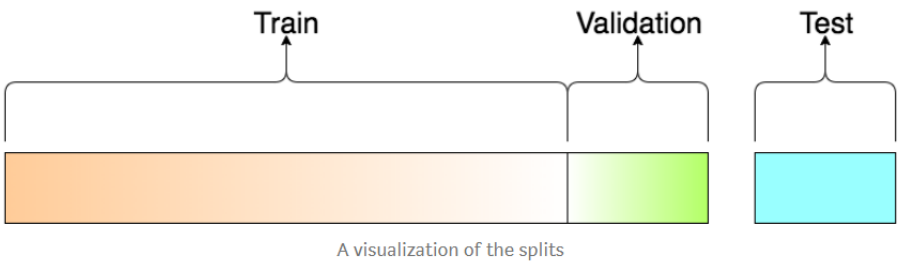
4.3.6 SW mimarisi çıktıları

Bu alt başlık kapsamında ortamdaki MCUXpresso ortamında yazılım tasarımı akış diyagramı ve yazılım tasarlanırken oluşturulan katmanlar üzerinde durulmuştur. Bu katmanlar detaylıca açıklanmaya çalışılmış ve yapılan çalışmalar görsellerle zenginleştirilmiştir.

5. DENEYSEL SONUÇLAR ve TARTIŞMA

Tez kapsamında bir gömülü sistem mimarisi üzerinden kullanıcıdan bağımsız bir şekilde tanımlanmış komutların yüksek başarı ile algılanması hedeflenmiştir. Bu kapsamda yapılan araştırmalar neticesinde çalışmalara PC ortamında başlanmış; sinyal işleme üzerine detaylı araştırma yapılmış, ardından Makine öğrenmesi üzerine çalışmalar gerçekleştirilmiştir. Yapılan çalışmaların sonucu olarak derin öğrenme yöntemi olan Convolutional Neural Network (CNN) ve Recurrent Neural Network (RNN) mimarilerinde modeller Google Colab platformunda elde edilmiştir. Çalışmalar sonucunda elde edilen bu modeller TensorFlow kütüphanesi sayesinde low level programlamaya indirgenerek DSP işlemleri yapma kapasitesi olan ve TensorFlow Lite yapısını destekleyen i.MX RT 1060 işlemcisi üzerinde gerçeklenmiş ve test edilmiştir. Gömülü işlemci mimrarisine geçişte üretici firma olan NXP Semiconductors'un sağlamış olduğu datasheet ve user manual vb. dökümanlar sayesinde ilgili mimari için yazılım geliştirilmiştir. Sistem eğitimi için kullandığımız dataset 3 ana başlık altında incelenmektedir: Eğitim veri seti, modeli eğitmek için kullandığımız gerçek veri kümesi (Sinir Ağı durumunda ağırlıklar ve sapmalar). Model bu verileri görür ve öğrenir. Validasyon veri seti yani Dev set, validasyon seti belirli bir modeli değerlendirmek için kullanılır, ancak bu sık değerlendirme içindir. Validasyon seti sonuçları, daha yüksek seviye hiperparametreleri ince ayarla güncellemek için kullanılır. Keras, derin bir öğrenme modeli eğitimi sırasında her bir eğitim aşamasının kaydedilmesini sağlar. Tüm derin öğrenme modellerini eğitirken kaydedilen varsayılan eğitim aşamalarını elde edebiliriz. Her eğitim aşaması için eğitim metriklerini kaydeder. Bu, eğer ayarlanmışsa, doğrulama veri kümesinin kaybı ve doğruluğunu (sınıflandırma problemleri için) içerir. Geçmiş nesnesi çağrılardan, modeli eğitmek için kullanılan fit () işlevine döndürülür. Metrikler, döndürülen nesnenin geçmiş üyesindeki bir sözlükte saklanır. Modeli keras olarak eğitirken, doğrulama verileri için keras modelindeki doğruluk ve kayıp farklı vakalarda değişiklik gösterebilir. Genellikle iterasyon sayısı arttıkça kayıplar azalmalı ve doğruluk artmalıdır. Bir makine öğrenme modeli eğitirken, kaçınmak istenilen ana

şeylerden biri aşırı uyum sağlamak olacaktır. Bu, modelin egzersiz (train) verilerine iyi uyduğu zamandır, ancak daha önce görmediği veriler (validasyon verileri) için genelleme yapamaz ve doğru tahminler yapamaz. Modellerinin gereğinden fazla uyup uymadığını öğrenmek için, veri bilimcileri çapraz doğrulama adı verilen bir teknik kullanırlar, burada verilerini iki kısma ayırırlar; eğitim seti ve validasyon seti. Eğitim seti modeli eğitmek için kullanılırken, validasyon seti, eğitim setinin dışındaki bilinmeyen veriler alınarak, yalnızca modelin performansını değerlendirmek için kullanılır. Eğitim setindeki metrikler, modelinizin eğitimi açısından nasıl ilerlediğini görmenizi sağlar, ancak validasyon setindeki verilerle, modelinizin kalitesini ölçmenize olanak tanıyan metrikler verilere dayalı yeni tahminler yapabilmeyi sağlar. Validasyon_kaybı (val_loss) ve validasyon_doğruluk (val_acc) ise validasyon setindeki kayıp ve doğruluk ölçümleridir. EK 5-A'da Colab üzerinden alınmış CNN model özeti görseli verilmiştir. Çapraz Doğrulama (Cross Validation)**,** validasyon, için çapraz doğrulama yöntemini kullanabiliriz. Literatürde çeşitli çapraz doğrulama yöntemleri vardır. Birinci deney için yukarıda olduğu gibi veri setinin yüzde 20'lik kısmını validasyon için yüzde 80'lik kısmını eğitim için ayırıyoruz. Bu durum şekil 5.1'de görülmektedir:



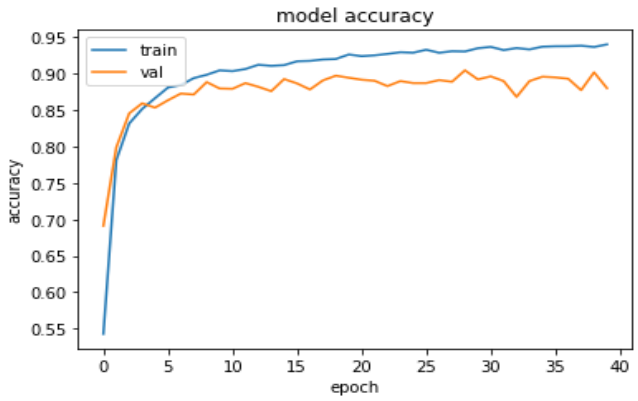
Şekil 5.1 : Veri Seti dağılımı.

İkinci deney için veri setinin farklı bir yüzde 20'lik kısmını test için yüzde 80'lik kısmını eğitim için ayırıyoruz ve bu prosedürü 3. 4. 5. deneyler için de gerçekleştirmekteyiz. En sonunda modellerin başarı oranlarının ortalamasını alıyoruz. Bizim test sonuçlarımız validasyon sonuçlarına çok yakın çıkmıştır.

Çizelge 5.1: CNN Colab Eğitim Sonuçları.

Kayıp	Doğruluk	Test_kaybı	Test_Doğruluk
0.3050	0.9351	0.4671	0.8923
0.3014	0.9369	0.4664	0.8964
0.3026	0.9325	0.4846	0.8900
0.3067	0.9353	0.5557	0.8962
0.2942	0.9335	0.4754	0.8960
0.2931	0.9372	0.4875	0.8951
0.2889	0.9379	0.4484	0.8932
0.2989	0.9385	0.4359	0.8775

LSTM mimarisi geliştirilerek tasarlanan deep learning modelinin için analiz çizelgeleri ve training history bilgileri gösterilmektedir. Colab üzerinde LSTM yöntemi kullanılarak 100 tur adımı (epoch) sonucu elde edilmiş eğitim sürecinin son 4 adımını göstermektedir. Şekil 5.2'de ve çizelge 5.2'de CNN model doğruluk grafiği ve eğitim sonuçları görülmektedir.

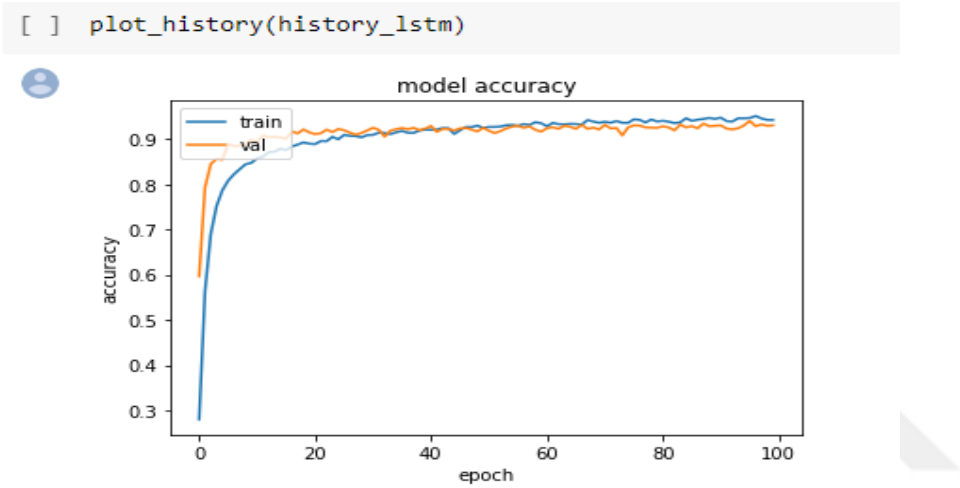


Şekil 5.2: CNN Model Özeti.

Çizelge 5.2: LSTM Colab Eğitim Sonuçları.

Kayıp	Doğruluk	Test_kaybı	Test_Doğruluk
0.1326	0.9508	0.6310	0.9282
0.1359	0.9454	0.6762	0.9322
0.1607	0.9421	0.6034	0.9291
0.1706	0.9418	0.5276	0.9300

Verilen sonuçlar eğitim ve validasyon setleri üzerinden yapılmıştır. Burada verilen kayıp değerleri bir maliyet fonksiyonun aldığı sürekl bir değer olduğu için göreceli olarak yüksektir. Bunun nedeni doğruluk hesaplanırken düzeltme veya yuvarlama sonucu elde edilen ayrık değişken kullanılır (örneğin 0.7 olan ortalama bir değeri eşikten büyük olduğu için 1'e yuvarlıyoruz. Aradaki hata kümülatif olarak kayıp değerlerine eklendiği için kayıp değerleri artmaktadır. Ayrıca burada doğrulukta sağlanmaktadır). LSTM yönteminde kayıplar artmaktadır çünkü birkaç adım ötedekileride hesaplamak için her aşamada hatalar eklenerek büyümektedir. Şekil 5.3'de LSTM model grafik özeti görülmektedir.



Şekil 5.3: LSTM Model Özeti.

Keras, derin bir öğrenme modeli eğitimi sırasında her bir eğitim aşamasının kaydedilmesini sağlar. Tüm derin öğrenme modellerini eğitirken kaydedilen varsayılan eğitim aşamalarını elde edebiliriz. Her eğtim aşaması için eğitim metriklerini kaydeder. Bu, eğer ayarlanmışsa, doğrulama veri kümesinin kaybı ve doğruluğunu (sınıflandırma problemleri için) içerir. Geçmiş nesnesi çağrılardan, modeli eğitmek için kullanılan fit () işlevine döndürülür. Metrikler, döndürülen nesnenin geçmiş üyesindeki bir sözlükte saklanır. Modeli keras olarak eğitirken, doğrulama verileri için keras modelindeki doğruluk ve kayıp farklı vakalarda değişiklik gösterebilir. Genellikle her çağ arttıkça kayıplar azalmalı ve doğruluk artmalıdır. Bir makine öğrenme modeli eğitirken, kaçınmak istenilen ana şeylerden biri aşırı uyum sağlamak olacaktır. Bu, modelin egzersiz (train) verilerine iyi uyduğu zamandır, ancak daha önce görmediği veriler için genelleme yapamaz ve doğru tahminler yapamaz.

Modellerinin gereğinden fazla uyup uymadığını öğrenmek için, veri bilimcileri çapraz doğrulama adı verilen bir teknik kullanırlar, burada verilerini iki kısma ayırırlar - eğitim seti ve doğrulama seti. Eğitim seti modeli eğitmek için kullanılırken, doğrulama seti yalnızca modelin performansını değerlendirmek için kullanılır. Eğitim setindeki metrikler, modelinizin eğitimi açısından nasıl ilerlediğini görmenizi sağlar, ancak doğrulama setindeki, modelinizin kalitesini ölçmenize olanak tanıyan metrikler-verilere dayalı yeni tahminler yapabilmeyi sağlar. Bu noktada, görüleceği üzere kayıp ve doğruluk terimleri eğitim setindeki kayıp ve doğruluk ölçümleridir, test_kaybı (val_loss) ve test_doğruluk (val_acc) ise doğrulama setindeki kayıp ve doğruluk ölçümleridir. EK 5-A'da Colab üzerinden alınmış model özeti görseli verilmiştir.

Çizelge 5.3'de CNN Colab model parametreleri yer almaktadır.

Çizelge 5.3: CNN Model Parametreleri

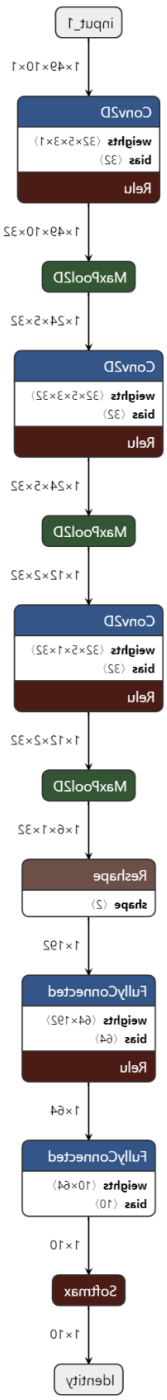
Kayıp	Doğruluk	Test_kaybı
Input Layer	(None, 49, 10, 1)	0
Conv2D	(None, 49, 10, 32)	512
Activation	(None, 49, 10, 32)	0
Max_pooling2D	(None, 24, 5, 32)	0
Conv2d_1	(None, 24, 5, 32)	15392
Max_pooling2D_1	(None, 24, 5, 32)	0
Conv2d_2	(None, 12, 2, 32)	0
Activation_2	(None, 12, 2, 32)	5152
Max_pooling2D_2	(None, 6, 1, 32)	0
Flatten	(None, 192)	0
Dense	(None,64)	12352
Dense_3	(None, 10)	650

Derin öğrenme veya makine öğrenme modelini oluşturma sürecini tamamladıktan sonra, çıkarım işleminize uygulamadan önce modeli görselleştirmek ve daha iyi anlayabilmek için Netron programını kullanıldı. Bu program sayesinde modelin girdileri, çıktıları, katman mimarisi ve kullanılan aktivasyon fonksiyonlarını detaylıca görebilmekteyiz. Netron model görselleştirme uygulaması OSX, Linux ve Windows OS için yürütme desteğine sahiptir [45]. Çizelge 5.4'de Ram/Rom optimizasyonu görülmektedir. Ayrıca çizelge 5.5'de LSTM Colab model parametreleri verilmiştir.

Çizelge 5.4: Gömülü Mimari RAM / ROM Optimizasyonu

Memory Region	Used Size	Region Size	% age used
Board_Flash	1984700 B	8 MB	23.66 %
SRAM_DTC	118680 B	128 KB	90.55 %
SRAM_ITC	0 GB	128 KB	0.00 %
SRAM_OC	627920 B	768 KB	79.84 %

Şekil 5.4’de elde edilen modelin netron programı görselleştirilmiş hali verilmiştir.



Şekil 5.4: CNN Netron ile Görselleştirme.

Çizelge 5.5: LSTM Colab Model Parametreleri

Katman (tip)	Çıktı Verisi	Parametre #
lstm1 (LSTM)	(NONE, 256)	273408
Dense_32 (Dense)	(NONE, 64)	16488
Dropout_11 (Dropout)	(NONE, 64)	0
Dense_33 (Dense)	(NONE,10)	650

LSTM, uzun dönemli ilişkisel korelasyon verilerini dikkate aldığı için veri girişleri homojen ve ardışıl (sequential) olmalıdır. CNN model de ise,veri seti girişleri heterojen ve bağımsız olabilir. Veri girişindeki bu farklılık LSTM modelin önceki ve sonraki veri arasında bağ kurmasına ve çok daha yüksek başarım oranı elde etmesine neden olur. Ayrıca LSTM'de katman sayısı çok daha azdır. Bu yüzden ara kademelerde veya katmanlarda görece daha uzun blok verilerle çalışırız. Öte yandan CNN modelinde çok zaman alan ek budama (pruning – pooling) katmanlarına gerek kalmamaktadır. LSTM modelinde daha hızlı bir şekilde ara kademeye geçilebilmektedir.

Tez kapsamında hazırlanan proje yapılan çalışma sonucu elde edilen flash ve ram kullanım oranları gösterilmiştir. EK 5-B'de prrogram compiler çıktısı görülmektedir.

Programın çalıştırılması aşağıdaki işlemler takip edilerek tamamlanır:

NXP MCUxpresso ide'si üzerinden yazılan gömülü işlemci mimarisi açılır ve MIMX RT 1060 EEval Board usb kablosu üzerinden PC'ye takılır.

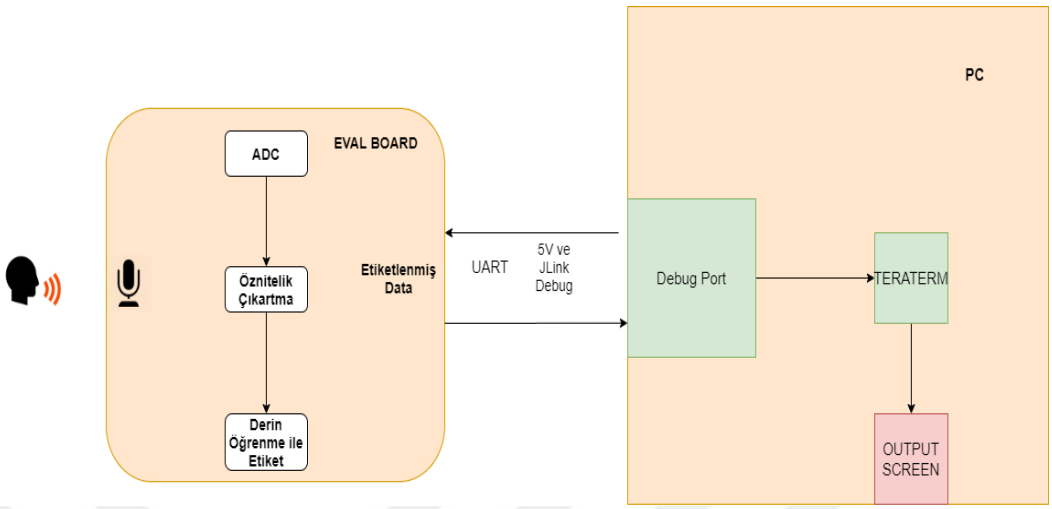
GCC compiler altyapısana sahip ide ile gömülü işlemci yazılımı derlenir ve debugger üzerinden PC'ye bağlı olan geliştirme kartına atılır.

Kod debug moda getirilir ve TeraTerm programı açılır.

Debug modun başlatılmasıyla birlikte geliştirme kartı üzerinde ki mikrofon data almaya başlar ve wakeup word'ün gelmesini bekler. Wakeup word'ün gelmesiyle paralel koldaki AI model devreye girer ve komut bekler.

Gelen komutun statüsüne ve algılama oranına göre uart üzerinden PC'ye aktarılır ve TeraTerm üzerinden ekrana bastırılır. Şekil 5.5'de yukarıda verilen maddeler görsel

olarak verilmiştir. Keras LSTM eğitim verileri EK 5 – C'de ulaşılabilmektedir. Çizelge 5.6'da veri seti test sonuçları yer almaktadır.



Şekil 5.5: Basitleştirilmiş Çalışma Mekanizması.

Çizelge 5.6: Veri Seti Kullanıcı Test Sonuçları

Kelime	Başarım Oranı
Bed	%94
Eight	%90
Five	%86
Off	%94
On	%95
Right	%93
Sheila	%92
Silence (Sessizlik)	%99
Unkown Kateg.	%91
Zero	%90

Tez kapsamında ele alınan veriseti içerisinde yer alan kelime listesinin başarım oranlarını içermektedir. Bu sonuç listesi İstanbul Türkçesi ile konuşan bir kadın, iki erkeğin önce wakeup Word ardından her bir komutu 10 defa söylemesi sonucu elde edilmiş yuvarlanmış sonuçlardır. Bu kapsamda çıkan sonuçların değerlendirmesi yapılırken; datasetin anadili İngilizce konuşmacıların olduğunun bilinmesi, çevre gürültü faktörlerinin ve gömülü mimaride ki yaşanabilecek performans kayıpları göz önüne alınması gerekmektedir.

6. SONUÇ VE ÖNERİLER

Yapılan çalışmaları sırasıyla özetlemek gerekirse; ilk başta yapılan çalışma yazılımsal olarak sinyal işleme algoritmaları üzerinde duruldu. Devamında ilgili algortimalar kullanılarak analog sinyallarin sayısallaştırılması işlemleri gerçekleştirildi. Elde edilen bu sonuçlar sayısal özniteliklerini kullanacağımız sayısal sinyali elde etmek konusunda son derece önemli rol oynadı. Devamında ise yapılan çalışmalar kapsamında MFCC (Mel Frekansı Septral Katsayıları) yöntemi kullanılarak; derin öğrenme modelimize giriş kümesini sağlayacak olan mel spektrogram matrisinin elde edilmesi sağlandı. Bu aşamadan sonraki süreç ise bu elde edilen öznitelik vektör matrisinin eğitimde kullanılması ve mikrofon vasıtasıyla alınacak olan verinin, doğru şekilde etiketlenmesi aşaması oldu. Bu noktada Pyton, Keras, TensorFlow ve TFLite gibi değerli programlama dili ve kütüphaneler kullanıldı.

Derleme/Benzetim ortamı olarak ise verdiği GPU desteği sayesinde hızlı eğitim olanağı sağlayan Google Colab tercih edildi. Modelin elde edilmesi sonucunda geriye kalan nokta sinyal işleme yazılımlarını ve modeli gömülü sistem mimarisine entegre etmeye gelmişti. Bunlara ek olarak iyi bir mikrofonun seçimide en önemli kriterlerden bir tanesiydi. Yapılan araştırmalar sonucu NXP RT1060'da karar kılındı. Hem sağladığı işlemci hızı, açık kütüphaneleri ve örnek çalışmaları ile son derece değerli kaynaklar sundu.

Yapılan çalışmanın gömülü işlemci mimarisine aktarılmasının ardından; gömülü sistemler ile PC çalışmaları kıyaslandığında performans kayıpları ortaya çıkmıştır. Sebepler üç ana kategoride gruplandırılabilir: ses yalıtımı problemleri, kullanıcı hataları ve çevresel etkilerdir. Tanıma ortamındaki gürültü, hem konuşmanın spektral özelliklerini deforme ederek hem de uç noktaların yanlış algılanmasına neden olarak tanıma performansını azaltır. Tanıma yapılan odanın akustik özellikleri, konuşmanın spektral özelliklerini de etkiler.

Bu problemler ışığında yapılacak olan ileri dönem çalışmalar daha geniş bir dataset, speech direction detection, active noise filtering, voice childlock, cihazın bulunduğu

ortamının akustik özelliklerini öğrenmesi, speech komutlarının bluetooth mesh ağlarına bağlı cihazlar üzerinden birbirine aktarılması gibi konular üzerinde durulacaktır. Komut tanıma ortamındaki gürültü, hem konuşmanın spektral özelliklerini deforme ederek hem de uç noktaların yanlış algılanmasına neden olarak tanıma performansını azaltır. Tanıma yapılan odanın akustik özellikleri, konuşmanın spektral özelliklerini de etkiler. Odanın akustik özellikleri biliniyorsa, sayısal filtre kullanmak yansıma etkisine bir çözüm olabilir.

LSTM, uzun dönemli ilişkisel korelasyon verilerini dikkate aldığı için veri girişleri homojen ve ardışıl (sequential) olmalıdır. CNN model de ise, veri seti girişleri heterojen ve bağımsız olabilir. Veri girişindeki bu farklılık LSTM modelin önceki ve sonraki veri arasında bağ kurmasına ve çok daha yüksek başarım oranı elde etmesine neden olur. Ayrıca LSTM'de katman sayısı çok daha azdır. Bu yüzden ara kademelerde veya katmanlarda görece daha uzun blok verilerle çalışırız. Öte yandan CNN modelinde çok zaman alan ek budama (pruning – pooling) katmanlarına gerek kalmamaktadır. LSTM modelinde daha hızlı bir şekilde ara kademeye geçilebilmektedir. İlerideki çalışmalarda LSTM TensorFlow 2.3'den itibaren gömülü sisteme aktarılacaktır.

KAYNAKLAR

[1] Meral M., (2006) EMBEDDED SPEECH RECOGNITION, Master thesis, İstanbul Teknik Üniversitesi.

[2] Ruiz B. C., Tapias g. E. F., (2005) Microcontroller Implementation of a Voice Command Recognition System for Human-Machine Interface in Embedded Systems.

[3] Zaman S., Slany W., (2014) " Smartphone-Based Online and Offline Speech Recognition System for ROS-Based Robots.

[4] Nedevschi S., Patra, R. K., Brewer E., (2005) Hardware Speech Recognition for User Interfaces in Low Cost, Low Power Device.

[5] Macial A. M. A., Veiga A., Lopes J., (2008) A ROBUST SPEECH COMMAND RECOGNIZER FOR EMBEDDED APPLICATIONS.

[6] Erdemir C., (2010) TÜRKÇE METİN SESLENDİRME İÇİN DOĞAL KONUŞMA SENTEZLEME, Master Tezi, İstanbul Üniversitesi.

[7] Rabiner L., Juang B., (1993). Fundamentals of Speech Recognition, Prentice Hall.

[8] Baygün K. M., (2006) TÜRKÇE KOMUTLARI TANIYAN SES TANIMA SİSTEM GELİŞTİRİLMESİ, Master Tezi, Pamukkale Üniversitesi.

[9] Greenstreet D., Smsrtik J., (2017) Voice as the user interface – a new era in speech processing, Texas Instruments.

[10] EMO, **Elektronik Notları**. **(2013)**1.bs.--Ankara. Elektrik Mühendisleri Odası, 136s. : 24 cm (EMO Yayın No: EK/2013/546; ISBN: 978-605-01-0421-5).

[11] Xiaofan L., Fangwei D., (2019) A Survey on Deep Learning Techniques in Wireless Signal Recognition.

[12] Arnone L., Bocchio S., Rosti A., (2004). On embedded system architectures for speech recognition applications: the gap between the status and the demand .

[13] Uslu G. Z., (2019) SESLİ TIBBİ RAPORLARIN YAZIYA DÖNÜŞTÜRÜLMESİ, Master Tezi, Yıldız Teknik Üniversitesi.

[14] Çakır M. Y., (2017) Gerçek Zamanlı Yüksek Kalitede Ses Tanıma, Master Tezi, İstanbul Zaim Üniversitesi .

[15] Kai F. L., (1999) Automatic Speech Recognition, 4.th edition, Spinger Science, LLC.

[16] Marshall D., (2001), Discrete Cosine Matrix, University of Oxford, UK.

[17] Sen S., Dutta A., Dey N., (2018) Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews.

[18] Watanabe S., Delcroix M., (2017) New Era for Robust Speech Recognition: Exploiting Deep Learning, Spinger.

[19] Gokani J., (2017) The evololution of banking: AI, Stanford University

[20] Manaswi K. W., (2018) Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras, APress

[21] Francois C., (2017) Deep Learning with Pyton 1st edition ,Mannig Publish.

[22] Sharma S., (2017), Activation Functions in Neural Networks.

[23] Kandi, N., Khorasani, K., Patel, R. V., Sood, V. K. (1993) Optimum learning rate for backpropagation neural Networks. Canadian Conference on Electrical and Computer Engineering.

[24] Sinha P., (2010) Speech Recognition in Embedded Systems, Spinger, USA.

[25] Fridman L., (2019), Introduction and Overview with TensorFlow.

[26] Deng L. Y. D., (2015), London Automatic Speech Recognition A Deep Learning Approach, Spinger.

[27] Gudikandula P., (2019) Recurrent Neural Networks and LSTM explained.

[28] Zhang A., Lipton C. Z., Mu Li, Smola J., (2020) Dive into Deep Learning.

[29] Dell'Aversana P., (2019), ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING. A SIMPLE OVERVIEW, Milano.

[30]Url-1<https://www.nxp.com/design/software/development-software/eiq-ml-development-environment/eiq-for-tensorflow-lite:eIQTensorFlowLite >, erişim tarihi 07.05.2020.

[31]Url-2<https://www.nxp.com/applications/solutions/enabling-technologies/ai-and-machine-learning:MACHINE-LEARNING >, erişim tarihi 03.05.2020.
[32]Url-3<https://www.nxp.com/design/software/development-software/eiq-ml-development-environment:EIQ>, erişim tarihi 16.04.2020.

[33] Geron A., (2017) Hands-On Machine Learning with Scikit-Learn and TensorFlow.

[34] Ghasemian A., Hosseinmardi H., Clauset A., (2019) Evaluating Overfit and Underfit in Models of Network Community Structure.

[35]Url-4 < https://keras.io/ >, erişim tarihi 19.05.2020.

[36]Url-5<https://www.nxp.com/design/training/simplifying-deep-learning-and-neural-networks-for-embedded-processing:TIP-SIMPLIFYING-DEEP-LEARNING >, erişim tarihi 30.05.2020.

[37]Url-6 < https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/i-mx-rt-crossover-mcus/i-mx-rt1060-crossover-mcu-with-arm-cortex-m7-core:i.MX-RT1060>, erişim tarihi 01.06.2020.

[38]Url-7< https://www.arm.com/ >, erişim tarihi 02.06.2020.

[39] Arnold J., (2017) An Introduction to Floating-Point Arithmetic and Computation, CERN.

[40] Hain T., Marino D., (2011) An Analysis of Automatic Speech Recognition with Multiple Microphones.

[41] **Url- 8** < https://www.cirrus.com/>, erişim tarihi 02.06.2020.

[42] **Url -9** < https://www.cirrus.com/products/wm8960/>, erişim tarihi 02.06.2020.

[43]**Url-10**<https://www.nxp.com/docs/en/application-note/AN12437.pdf>, erişim tarihi 03.06.2020.

[44] Wagner P., Li L., Wild T., Mayer A., Herkersdorf A., Knowledge-Based On-Chip Diagnosis for Multi-Core Systems-on-Chip, München Germany.

[45] Gharibi G., Walunj V., Alanazi R., Rella S., Lee Y. (2019) Automated Management of Deep Learning Experiments.

EKLER

EK 4-A: Pencereleme fonksiyonun gömülü sistem mimarisinde koda dökülmesi.

```
// create window function
window_func = new float[frame_len];
for (int i = 0; i < frame_len; i++)
  window_func[i] = 0.5 - 0.5 * cos(M_2PI * ((float)i) / (frame_len));
```

EK 4-B: Normalizayon işleminin gömülü sistem mimarisinde koda dökülmesi.

```
// TensorFlow way of normalizing .wav data to (-1, 1)
for (i = 0; i < frame_len; i++) {
  frame[i] = (float)(audio_data[i]) * 1.0 / (1 << 15);
}
```

EK 4-C: FFT ve Power Specktrum işlemlerinin gömülü sistem mimarisinde koda dökülmesi.

```
// Compute FFT
arm_rfft_fast_f32(rfft, frame, buffer, 0);

// Convert to power spectrum
// frame is stored as [real0, realN/2-1, real1, im1, real2, im2, ...]
int32_t half_dim = frame_len_padded / 2;
float first_energy = buffer[0] * buffer[0],
      last_energy =  buffer[1] * buffer[1];  // handle this special case
for (i = 1; i < half_dim; i++) {
  float real = buffer[i * 2], im = buffer[i * 2 + 1];
  buffer[i] = real * real + im * im;
}
buffer[0] = first_energy;
buffer[half_dim] = last_energy;

float sqrt_data;
```

EK 4-D: MEL Filtre Bankası uygulamasının gömülü sistem mimarisinde koda dökülmesi.

```
// Apply mel filterbanks
for (bin = 0; bin < NUM_FBANK_BINS; bin++) {
  j = 0;
  float mel_energy = 0;
  int32_t first_index = fbank_filter_first[bin];
  int32_t last_index = fbank_filter_last[bin];
  for (i = first_index; i <= last_index; i++) {
    arm_sqrt_f32(buffer[i], &sqrt_data);
    mel_energy += (sqrt_data) * mel_fbank[bin][j++];
  }
  mel_energies[bin] = mel_energy;

  // avoid log of zero
  if (mel_energy == 0.0)
    mel_energies[bin] = FLT_MIN;
}
```

EK 4-F: DCT Matris Uygulamasının gömülü sistem mimarisinde koda dökülmesi.

```
//Take log
for (bin = 0; bin < NUM_FBANK_BINS; bin++)
  mel_energies[bin] = logf(mel_energies[bin]);

//Take DCT. Uses matrix mul.
for (i = 0; i < num_mfcc_features; i++) {
  float sum = 0.0;
  for (j = 0; j < NUM_FBANK_BINS; j++) {
    sum += dct_matrix[i*NUM_FBANK_BINS+j] * mel_energies[j];
  }
  mfcc_out[i] = sum;
}
```

EK 4-G : RuınInference Fonksiyonu Elde Edilen Datanın Etiketlenmesi.

```
  if (!top_results.empty())
  {
      auto result = top_results.front();
      const float confidence = result.first;
      const int index = result.second;
      if (
          ( confidence * 100 > DETECTION_TRESHOLD )
/         &&
/         ( index != 5 )
          &&
          (sample_cnt > 0)
  )
  {
    LOG(INFO) << "---\r\nDetected: " << std::setw(10) << labels[index] << " (" << (int)(confidence * 100) << "%)\r\n---\r\n\r\n";

/   LOG(INFO) << "----------------------------------------\r\n";
/   LOG(INFO) << "     Inference time:   " << (end - start) / 1000 << " ms\r\n";
/   LOG(INFO) << "     Detected: " << std::setw(10) << labels[index] << " (" << (int)(confidence * 100) << "%)\r\n";
/   LOG(INFO) << "----------------------------------------\r\n\r\n";
  }
  }
```

EK 5-A: Keras CNN Model Özeti.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 49, 10, 1)]	0
conv2d (Conv2D)	(None, 49, 10, 32)	512
activation (Activation)	(None, 49, 10, 32)	0
max_pooling2d (MaxPooling2D)	(None, 24, 5, 32)	0
conv2d_1 (Conv2D)	(None, 24, 5, 32)	15392
activation_1 (Activation)	(None, 24, 5, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 12, 2, 32)	0
conv2d_2 (Conv2D)	(None, 12, 2, 32)	5152
activation_2 (Activation)	(None, 12, 2, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 1, 32)	0
flatten (Flatten)	(None, 192)	0
dense (Dense)	(None, 64)	12352
dense_3 (Dense)	(None, 10)	650

Total params: 34,058
Trainable params: 34,058
Non-trainable params: 0

EK 5-B: MCUXPRESSO Gömülü sistem mimarisi.

```
Memory region         Used Size  Region Size  %age Used
     BOARD_FLASH:     1984700 B         8 MB     23.66%
        SRAM_DTC:      118680 B       128 KB     90.55%
        SRAM_ITC:          0 GB       128 KB      0.00%
         SRAM_OC:      627920 B       768 KB     79.84%
Finished building target: evkmimxrt1060_tensorflow_lite_kws.axf
```

EK 5-C: Keras LSTM Model Özeti.

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm1 (LSTM)                 (None, 256)               273408
_________________________________________________________________
dense_32 (Dense)             (None, 64)                16448
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_33 (Dense)             (None, 10)                650
=================================================================
Total params: 290,506
Trainable params: 290,506
Non-trainable params: 0
_________________________________________________________________
```

ÖZGEÇMİŞ

Ad-Soyad : Can ÇETİN

Doğum Tarihi ve Yeri : İstanbul – 24.05.1994

E-posta : can.cetin@outlook.com.tr

ÖĞRENİM DURUMU

- **Lisans :** 2017, İstanbul Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği (GPA: 3.52 / 4.00)

MESLEKİ DENEYİM VE ÖDÜLLER

- **MT Bilgi Teknolojileri ve Dış Tic. AŞ**, Gömülü Sistem Mühendisi
- **Arçelik AŞ,** Merkez Ar&Ge Gömülü Sistemler Tasarım Mühendisi