Emir Atak 2515518 Alkım Bozkurt 2515674

EE435 Communications I

Term Project Phase 1 Report

1. Introduction

In this project, classification of analog modulated radio signals according to their modulation type is performed using a deep learning method, ResNet. The intended classification system will be able to distinguish 5 modulation types. To be acquinted with methodology behind deep learning approaches to accomplish signal classification, literature is reviewed starting from provided paper [1]. To use ResNet for classification, it must be trained by synthetic datasets. To generate synthetic signals which constitutes datasets, signal model parameters are determined according to [1] at first. Then, generation of datasets is performed for each modulation type under various signal to noise ratio (SNR) and carrier frequency offset (CFO) values. Then the classification performance of the ResNet model will be evaluated in different SNR and CFO scenarios. In this report, literature review about radio signal classification is given at first. Then, dataset generation is explained with determined signal model parameters.

2. Literature Review

Classification of radio signals has been very important working area for numerous radio sensing and communication systems. The classification has been accomplished by carefully hand-crafting specialized feature extractors for many years. Hand-crafting specialized feature extracting is a manual process in which some algorithms or mathematical formulations are tried to create to classify the raw data by extracting its specific features. To accomplish this, deep knowledge about the data, its properties and expertise are needed. To handle the classification, deep learning based methods are being used in recent years. Deep Learning (DL) techniques often use raw data directly. Residual Network (ResNet) is a deep learning architecture using a concept called residual learning to make it easier to train networks with many layers, avoiding the issues of vanishing gradients and degradation of accuracy [2]. Neural networks like ResNet automatically learn relevant features during training. To train the network, a robust dataset must be generated [3]. The success of ResNet depends directly on the quality of the dataset. DL methods increased the capacity and pace for feature learning compared to the traditional methods like hand-crafted feature extractors.

3. Data Generation

In order to train our DL model, a comprehensive dataset is crucial to ensure the model learns to distinguish various modulation types effectively. The dataset is typically generated by simulating signals for the given modulation types namely DSB-SC, DSB-WC, SSB-SC, SSB-WC and FM. We implemented the channel impairments which were mentioned in the previous chapter for a more reliable simulation of real-world data transmission.

$$s(t) = m(t)_I \cdot cos(2\pi f_c ct) - m(t)_Q \cdot sin(2\pi f_c t) \qquad [1]$$

Equation above is the time-domain modulated signal when we neglect the effect of CFO. Ideally, we would obtain a complex signal where the message consists of in-phase and quadrature parts. For the given modulation types which we have generated signal, we constructed the message signal in the form of $m(t) = m_I(t) + jm_Q(t)$ to be able to utilize I/Q modulation. Our message signal is selected randomly within a set of audio files. We also randomly select 4096 samples from that audio file for modulation. Furthermore, we ensure the twice bandwidth constraint by up sampling the audio file.

Emir Atak 2515518 Alkım Bozkurt 2515674

Table 1: Modulated Signal s(t)

Modulation	**s(t)**
DSB-SC	$m(t)\cdot\cos(2\pi f_c t)$
DSB-WC	$[Ac + m(t)] \cdot \cos(2\pi f_c\ t)$
SSB-SC	$m(t)\cos(2\pi f_c t) \pm \hat{m}(t)\sin(2\pi f_c t)$
SSB-WC	$s(t)=Ac.\cos(2\pi f_c t)+[m(t)\cos(2\pi f_c t)\pm \hat{m}(t)\sin(2\pi f_c t)]$
FM	$Ac.\cos[2\pi f_c t+2\pi k_f \int m(\tau)d\tau]$

In real life, there is always a probability that there is a frequency deviation between the frequencies of carriers which are used to modulate and demodulate the message signal. This frequency deviation is defined as carrier frequency offset. We modelled the difference between the frequencies of carriers for modulation and demodulation Δf as gaussian process whose mean is 0 and standard deviation is 0.01 [1]. Because of the frequency deviation, message is not recovered perfectly. This can be shown in equations 2, 3 and 4. The carrier equation used for demodulation and equation for modulated signal is given in equations 2 and 3 respectively. The demodulated signal which is frequency shifted version of message signal is given in equation 4.

$$u(t) = cos(2\pi(f_c + \Delta f)t + \emptyset) \qquad [2]$$

$$s(t) = m(t) \cdot cos(2\pi f_c t) \qquad [3]$$

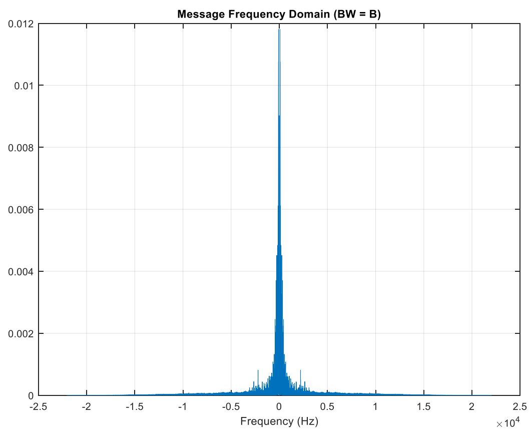
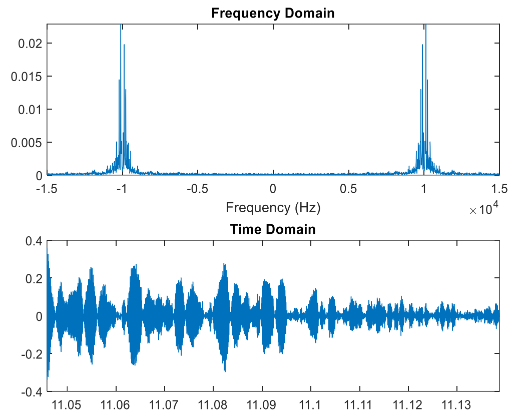
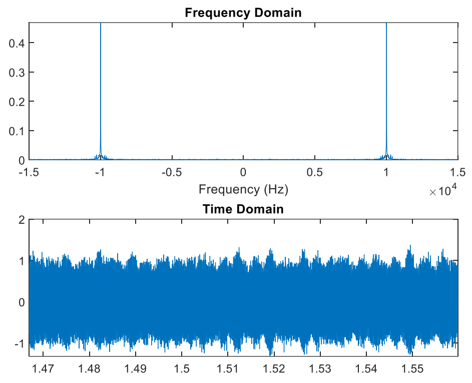
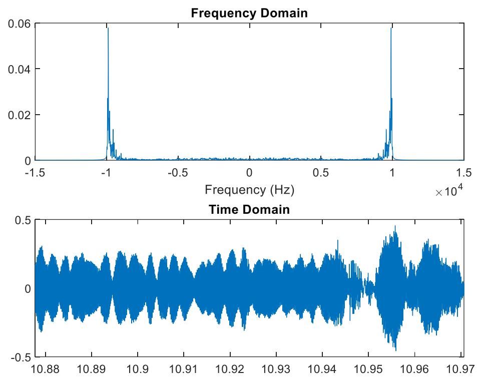
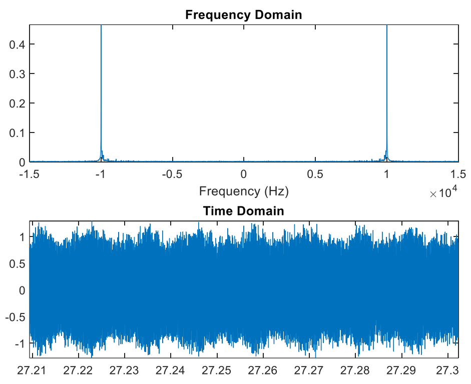
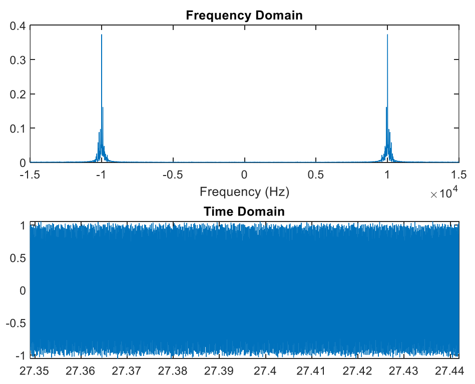
$$v(t) = m(t) \cdot cos(2\pi \Delta f t + \emptyset) \qquad [4]$$

To consider the thermal noise due to physical sensitivity of the devices, we simply implemented a AWGN process to our modulated signals. We have simulated several SNR scenarios from starting -20dB to 40 dB with 10dB steps, and we also simulated our channel under perfect noise (infinite SNR) conditions. In the below equation, r(t) is the received signal if we ignore path loss, and n(t) is AWGN process.

$$r(t) = s(t) + n(t) \qquad [5]$$

Although reference paper used GNU Radio for dataset generation [3], our approach was to utilize MATLAB as the simulation tool for modulating an arbitrary message signal and implement the random impairments. A sample of our dataset is a time domain signal consisting of 4096 samples with a fixed sample rate of 44.1 kHz.

Table 1: Generated Data

 **Message Signal**	 **DSB-SC**
 **DSB-WC**	 **SSB-SC**
 **SSB-WC**	 **FM**

4. Conclusion

In this project, a deep learning-based classification system was implemented using ResNet to distinguish between various analog modulated signals. Synthetic datasets were generated for five modulation types, incorporating varying SNR and CFO conditions, to simulate realistic communication scenarios. The ResNet model demonstrated its effectiveness in accurately classifying the signals under these conditions, highlighting the potential of deep learning approaches in radio signal classification. The findings provide valuable insights into leveraging residual networks for signal analysis and lay the groundwork for further exploration with real-world signal scenarios and hardware implementations.

5. References

[1] T. J. O'Shea, T. Roy and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," in IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168-179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv:1512.03385, 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1512.03385

[3] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in Proc. GNU Radio Conf., 2016, vol. 1, no. 1.

```
fs = 44100;
time = 0:1/fs:4095/fs;

SNR = [-20 -10 0 10 20 30 40 inf]; % Different SNR values

[message1, fs1] = audioread("donsen_2B_first.wav");
[message2, fs2] = audioread("donsen_B_first.wav");
[message3, fs3] = audioread("donsen_2B_second.wav");
[message4, fs4] = audioread("donsen_B_second.wav");
[message5, fs5] = audioread("hoscakal_2B_first.wav");
[message6, fs6] = audioread("hoscakal_B_first.wav");
[message7, fs7] = audioread("hoscakal_2B_second.wav");
[message8, fs8] = audioread("hoscakal_B_second.wav");
[message9, fs9] = audioread("ode_2B_first.wav");
[message10, fs10] = audioread("ode_B_first.wav");
[message11, fs11] = audioread("ode_2B_second.wav");
[message12, fs12] = audioread("ode_B_second.wav");
messages = {message1,message2,message3,message4,message5,message6,message7,message8,↙
message9,message10,message11,message12};

fs_s = [fs1,fs2,fs3,fs4,fs5,fs6,fs7,fs8,fs9,fs10,fs11,fs12];

number_samples = 4000;
data = zeros([number_samples 4096]);

fc = 10e3; %10kHz;
label = ones([number_samples 1]);

for sample_idx = 1:number_samples
    index = randi([1 12],1);
    % index = 6;
    message = messages{index};

    fs = fs_s(index);


    %CFO
    sigma = 0.01;  %variance for CFO
    X = randn(1);
    Y = sigma * X;


    delta_f_carrier = Y; %Carrier frequency offs    et ~N(0,sigma)

    fc_cfo_added = fc + delta_f_carrier;

    time = (0:length(message)-1)' / fs;
    carrier = cos(2*pi*(fc_cfo_added)*time); %carrier signal,


    segment_start = randi([1, length(message) - 4096]);  % Start of segment
    segment_end = segment_start + 4095;       % Random length between 5k and 50k↙
samples
    segment = message(segment_start:segment_end);
    new_carrier = carrier(segment_start:segment_end);
```

```matlab
    delta_f = randi([500 1000],1);

    %label:
    %0: DSB SC
    %1: DSB WC
    %2: SSB SC
    %3: SSB WC
    %4  FM

    label = label*4;
    % signal = dsb_sc_generator(segment,new_carrier);
    % signal = dsb_wc_generator(segment,new_carrier);
    % signal = ssb_sc_generator(segment,fc_cfo_added,fs);
    % signal = ssb_wc_generator(segment,new_carrier,fc_cfo_added,fs);
    signal = fm_generator(segment,delta_f,fc_cfo_added,fs);


    snr_index = randi([1,length(SNR)],1);
    noisy_signal = awgn(signal, SNR(snr_index), 'measured');

    data(sample_idx,:) = noisy_signal;


    time = time(segment_start:segment_end);
    %Frequency Domain
    subplot(2,1,1);
    L = length(noisy_signal);
    f=linspace(-fs/2,fs/2,L);
    Y=fftshift(fft(noisy_signal,L)/L);
    plot(f, abs(Y));
    xlim([-15e3 15e3]);
    title("Frequency Domain")
    xlabel("Frequency (Hz)")
    title("Frequency Domain")
    %Time Domain
    subplot(2,1,2)
    plot(time,noisy_signal)
    xlim([min(time) max(time)])
    title("Time Domain")
    % Save the noisy signal to a .mat file

end
filename = sprintf('fm.mat'); % Create filename
save(filename, 'data' ,'fs',"label");


function [dsb_sc] = dsb_sc_generator(message,carrier)
        dsb_sc = message.*carrier;
end

function [dsb_wc] = dsb_wc_generator(message,carrier)
        dsb_wc = (1+message).*carrier;
end
```

```
function [ssb_sc] = ssb_sc_generator(message,fc,fs)

        ssb_sc = ssbmod(message,fc,fs);
end

function [ssb_wc] = ssb_wc_generator(message,carrier,fc,fs)
        c = ssbmod(message,fc,fs);
        ssb_wc = c+carrier;
end

function [fm] = fm_generator(message, delta_f,fc,fs)

        fm = fmmod(message,fc,fs,delta_f);
end

function [linear_fm] = linear_fm_generator(kf)
        linear_fm = cos(2*pi*fc*time + pi*kf*time.*time);
end
```